Naive Bayes on Political Text

In this notebook we use Naive Bayes to explore and classify political data. See the `README.md` for full details.

In [1]:
```
import sqlite3
import nltk
import random
import numpy as np
from collections import Counter, defaultdict
import string
from nltk.corpus import stopwords

# Feel free to include your text patterns functions
# from text_functions_solutions import clean_tokenize, get_patterns
```

In [2]:
```
convention_db = sqlite3.connect("/Users/patriciomartinez/Downloads/2020_Conv
convention_cur = convention_db.cursor()
```

In [3]:
```
# List all tables in the database
tables = convention_cur.execute(
    "SELECT name FROM sqlite_master WHERE type='table';"
).fetchall()

tables
```

Out[3]:
```
[('conventions',)]
```

Part 1: Exploratory Naive Bayes

We'll first build a NB model on the convention data itself, as a way to understand what words distinguish between the two parties. This is analogous to what we did in the "Comparing Groups" class work. First, pull in the text for each party and prepare it for use in Naive Bayes.

In [4]:
```
convention_data = []

# fill this list up with items that are themselves lists. The
# first element in the sublist should be the cleaned and tokenized
# text in a single string. The second element should be the party.

query_results = convention_cur.execute(
    '''
    SELECT text, party
    FROM conventions
    '''
)

stopset = set(stopwords.words("english"))
```

```python
def clean_tokenize(text):
    text = text.lower()
    for p in string.punctuation:
        text = text.replace(p, " ")
    tokens = text.split()
    return [t for t in tokens if t not in stopset and t.strip()]

for row in query_results:
    raw_text = row[0]
    party    = row[1]
    tokens   = clean_tokenize(raw_text)
    cleaned  = " ".join(tokens)
    convention_data.append([cleaned, party])
```

Let's look at some random entries and see if they look right.

In [5]:
```python
random.choices(convention_data,k=10)
```

```
Out[5]: [['tougaloo college reflects progression people slavery citizenship scholar
        ship leadership contributing mississippi world alumni leaders like conventi
        on chairman congressman bennie thompson joe biden wants invest 70 billion h
        bcus like tougaloo imagine impact hbcus imagine impact hbcus could america
        mississippi cast 2 votes bernie sanders 38 votes next president joe biden',
          'Democratic'],
         ['could happen remi gideon 00 01 40 52 lebanon proud mother three children
        remi 01 40 57 speaks english arabic french earned degree psychology words c
        ould figure works daycare teacher virginia remi 01 41 10 says feel blessed
        loyal citizen greatest country world country given opportunity lifetime rea
        lize potential dreams remi 01 41 20 congratulations that's really great tha
        nk',
          'Republican'],
         ['democracy beautiful', 'Democratic'],
         ['look across aisle see party wants pursue dreams see democrat party wants
        dictate dreams don't see party wants free see party wants chain conformity
        destroy anyone deem heretic swore oath defend country constitution presiden
        t trump sworn that's he's advanced freedom despite savage political attacks
        overcome agenda radical left president trump unleashed economic might natio
        n like president history triggered rising tide working families brought us
        energy independence reclaimed jobs overseas democrats said would never retu
        rn fiercely defended besieged first second amendment start',
          'Republican'],
         ['racist coward … speaker 92 01 37 49 call … cops … speaker 93 01 37 49 go
        ing kill … speaker 94 01 37 49 i'm rape … speaker 95 01 37 58 mark mccloske
        y says family threatened violence',
          'Republican'],
         ['yeah', 'Democratic'],
         ['maryland… bianca shah 01 13 07 home frederick douglass… brandon scott 01
        13 09 …cast 1 vote bernie sanders 119 votes next president joe biden',
          'Democratic'],
         ['i'm congressman matt gaetz i'm speaking auditorium emptier joe biden's d
        aily schedule nation full hearts clear minds see choice clearly strength we
        akness energy confusion success failure president trump first president sin
        ce reagan start new war biden foolishly cheerled decades war without winnin
        g without end president trump knows strongest fight hardest distant deserts
        fellow americans must fight save america may lose forever joe biden might e
        ven notice settle biden that's hashtag promoted aoc socialists woketopians
        settle biden make extra movie written produced directed others it's horror
        film really they'll disarm empty prisons lock home invite ms 13 live next d
        oor',
          'Republican'],
         ['kayla wanted make home still working find god willing bring home kayla b
        orn miracle told would never second child god gave us kayla gave world eigh
        t months kayla's captivity another hostage smuggle letter kayla written rea
        d could see god holding arms words felt tenderly cradled freefall also wrot
        e "i shown darkness light learned even prison one free grateful many hours
        think absence finally 25 years old come realize place life none us could kn
        own would long know also fighting side ways able lot fight left inside brea
        king give matter long takes " marcia mueller 01 22 43 kayla taught many thi
        ngs mom she's still teaching us carl support donald trump commitment make k
        eep america great power government passion people like kayla americans even
        darkest days always fight left inside americans don't talk act daughter tha
        t's president trump long stay strong like kayla long refuse break great tha
        nk',
          'Republican'],
```

```
 ['joe biden’s america radical left get whatever want get pay they’ve alrea
dy taken joe biden democratic party don’t let take america',
  'Republican']]
```

If that looks good, we now need to make our function to turn these into features. In my solution, I wanted to keep the number of features reasonable, so I only used words that occur at least `word_cutoff` times. Here's the code to test that if you want it.

In [6]:
```
word_cutoff = 5

tokens = [w for t, p in convention_data for w in t.split()]

word_dist = nltk.FreqDist(tokens)

feature_words = set()

for word, count in word_dist.items() :
    if count > word_cutoff :
        feature_words.add(word)

print(f"With a word cutoff of {word_cutoff}, we have {len(feature_words)} as
```

```
With a word cutoff of 5, we have 2435 as features in the model.
```

In [7]:
```
def conv_features(text,fw) :
    """Given some text, this returns a dictionary holding the
       feature words.

       Args:
            * text: a piece of text in a continuous string. Assumes
            text has been cleaned and case folded.
            * fw: the *feature words* that we're considering. A word
            in `text` must be in fw in order to be returned. This
            prevents us from considering very rarely occurring words.

       Returns:
            A dictionary with the words in `text` that appear in `fw`.
            Words are only counted once.
            If `text` were "quick quick brown fox" and `fw` = {'quick','fox'
            then this would return a dictionary of
            {'quick' : True,
             'fox' :    True}

    """
    ret_dict = dict()
    for word in text.split():
        if word in fw:
            ret_dict[word] = True
    return ret_dict
```

In [8]:
```
assert(len(feature_words)>0)
assert(conv_features("donald is the president",feature_words)==
       {'donald':True,'president':True})
assert(conv_features("people are american in america",feature_words)==
                     {'america':True,'american':True,"people":True})
```

Now we'll build our feature set. Out of curiosity I did a train/test split to see how accurate the classifier was, but we don't strictly need to since this analysis is exploratory.

In [9]:
```
featuresets = [(conv_features(text,feature_words), party) for (text, party)
```

In [10]:
```
random.seed(20220507)
random.shuffle(featuresets)

test_size = 500
```

In [11]:
```
test_set, train_set = featuresets[:test_size], featuresets[test_size:]
classifier = nltk.NaiveBayesClassifier.train(train_set)
print(nltk.classify.accuracy(classifier, test_set))
```

```
0.494
```

In [12]:
```
classifier.show_most_informative_features(25)
```

```
Most Informative Features
                   china = True           Republ : Democr =     25.8 : 1.0
                   votes = True           Democr : Republ =     23.8 : 1.0
             enforcement = True           Republ : Democr =     21.5 : 1.0
                 destroy = True           Republ : Democr =     19.2 : 1.0
                freedoms = True           Republ : Democr =     18.2 : 1.0
                 climate = True           Democr : Republ =     17.8 : 1.0
                supports = True           Republ : Democr =     17.1 : 1.0
                   crime = True           Republ : Democr =     16.1 : 1.0
                   media = True           Republ : Democr =     14.9 : 1.0
                 defense = True           Republ : Democr =     14.0 : 1.0
                 beliefs = True           Republ : Democr =     13.0 : 1.0
               countries = True           Republ : Democr =     13.0 : 1.0
                    isis = True           Republ : Democr =     13.0 : 1.0
                 liberal = True           Republ : Democr =     13.0 : 1.0
                religion = True           Republ : Democr =     13.0 : 1.0
                   trade = True           Republ : Democr =     12.7 : 1.0
                    flag = True           Republ : Democr =     12.1 : 1.0
               greatness = True           Republ : Democr =     12.1 : 1.0
                 abraham = True           Republ : Democr =     11.9 : 1.0
                  defund = True           Republ : Democr =     11.9 : 1.0
                    drug = True           Republ : Democr =     10.9 : 1.0
              department = True           Republ : Democr =     10.9 : 1.0
               destroyed = True           Republ : Democr =     10.9 : 1.0
                   enemy = True           Republ : Democr =     10.9 : 1.0
               amendment = True           Republ : Democr =     10.3 : 1.0
```

Write a little prose here about what you see in the classifier. Anything odd or interesting?

My Observations

I notice that the “most informative” features list tells us which words are especially strong indicators for one party or the other. Whenever a speech includes the word "china", the model is about 26 times more likely to guess “Republican,” whereas the word "votes" makes it about 24 times more likely to predict “Democratic.” In general, words like "enforcement", "destroy", "freedoms", and "defense" tend to point toward Republicans, while "climate" and "votes" point toward Democrats.

Part 2: Classifying Congressional Tweets

In this part we apply the classifer we just built to a set of tweets by people running for congress in 2018. These tweets are stored in the database `congressional_data.db`. That DB is funky, so I'll give you the query I used to pull out the tweets. Note that this DB has some big tables and is unindexed, so the query takes a minute or two to run on my machine.

In [13]:
```
cong_db = sqlite3.connect("/Users/patriciomartinez/Downloads/congressional_c
cong_cur = cong_db.cursor()
```

In [14]:
```
results = cong_cur.execute(
        '''
           SELECT DISTINCT
                  cd.candidate,
                  cd.party,
                  tw.tweet_text
           FROM candidate_data cd
           INNER JOIN tweets tw ON cd.twitter_handle = tw.handle
               AND cd.candidate == tw.candidate
               AND cd.district == tw.district
           WHERE cd.party in ('Republican','Democratic')
               AND tw.tweet_text NOT LIKE '%RT%'
        ''')

results = list(results) # Just to store it, since the query is time consumir
```

In [15]:
```
tweet_data = []

# Now fill up tweet_data with sublists like we did on the convention speeche
# Note that this may take a bit of time, since we have a lot of tweets.

for candidate, party, tweet in results:
    cleaned_text = tweet.lower()
    tweet_data.append([cleaned_text, party])
```

There are a lot of tweets here. Let's take a random sample and see how our classifer does. I'm guessing it won't be too great given the performance on the convention speeches...

In [16]:
```
random.seed(20201014)

tweet_data_sample = random.choices(tweet_data,k=10)
```

In [17]:
```
for tweet, party in tweet_data_sample :
    # Fill in the right-hand side above with code that estimates the actual
    features = conv_features(tweet, feature_words)
```

```
    estimated_party = classifier.classify(features)

    print(f"Here's our (cleaned) tweet: {tweet}")
    print(f"Actual party is {party} and our classifer says {estimated_party}
    print("")
```

```
Here's our (cleaned) tweet: b'earlier today, i spoke on the house floor abt
protecting health care for women and praised @ppmarmonte for their work on t
he central coast. https://t.co/wqgtrzt7vv'
Actual party is Democratic and our classifer says Democratic.

Here's our (cleaned) tweet: b'go tribe! #rallytogether https://t.co/0nxutfl9
l5'
Actual party is Democratic and our classifer says Democratic.

Here's our (cleaned) tweet: b"apparently, trump thinks it's just too easy fo
r students overwhelmed by the crushing burden of debt to pay off student loa
ns #trumpbudget https://t.co/ckyqo5t0qh"
Actual party is Democratic and our classifer says Democratic.

Here's our (cleaned) tweet: b'we\xe2\x80\x99re grateful for our first respon
ders, our rescue personnel, our firefighters, our police, and volunteers who
have been working tirelessly to keep people safe, provide much-needed help,
while putting their own lives on the line.\n\nhttps://t.co/ezpv0vmiz3'
Actual party is Republican and our classifer says Democratic.

Here's our (cleaned) tweet: b'let\xe2\x80\x99s make it even greater !! #kag
\xf0\x9f\x87\xba\xf0\x9f\x87\xb8 https://t.co/y9qozd5l2z'
Actual party is Republican and our classifer says Democratic.

Here's our (cleaned) tweet: b"we have about 1hr until the @cavs tie up the s
eries 2-2. i'm #allin216 @repbarbaralee you scared? #roadtovictory"
Actual party is Democratic and our classifer says Democratic.

Here's our (cleaned) tweet: b'congrats to @belliottsd on his new gig at sd c
ity hall. we are glad you will continue to serve\xe2\x80\xa6 https://t.co/fk
vmw3cqdi'
Actual party is Democratic and our classifer says Democratic.

Here's our (cleaned) tweet: b'we are really close, we have over $3500 raised
toward the match right now. whoot!! (that\xe2\x80\x99s $7000 for the non-mat
h majors in the room \xf0\x9f\x98\x82). help us get there https://t.co/tu34c
472sd https://t.co/qsdqkypsmc'
Actual party is Democratic and our classifer says Democratic.

Here's our (cleaned) tweet: b'today, the comment period for @potus\xe2\x80\x
99s plan to expand offshore drilling opened to the public. you have 60 days
(until march 9) to share why you oppose the proposed program directly with t
he trump administration. comments can be made by email or mail. https://t.c
o/baaymejxqn'
Actual party is Democratic and our classifer says Democratic.

Here's our (cleaned) tweet: b'celebrated @icseastla\xe2\x80\x99s 22 years of
eastside commitment &amp; saluted community leaders at last night\xe2\x80\x9
9s awards dinner! https://t.co/7v7gh8givb'
Actual party is Democratic and our classifer says Democratic.
```

Now that we've looked at it some, let's score a bunch and see how we're doing.

In [18]:
```
# dictionary of counts by actual party and estimated party.
# first key is actual, second is estimated
```

```python
parties = ['Republican','Democratic']
results = defaultdict(lambda: defaultdict(int))

for p in parties :
    for p1 in parties :
        results[p][p1] = 0


num_to_score = 10000
random.shuffle(tweet_data)

for idx, tp in enumerate(tweet_data) :
    tweet, party = tp
    # Now do the same thing as above, but we store the results rather
    # than printing them.

    # get the estimated party
    features = conv_features(tweet, feature_words)
    estimated_party = classifier.classify(features)

    results[party][estimated_party] += 1

    if idx > num_to_score :
        break
```

In [19]:

```python
results
```

Out[19]:

```
defaultdict(<function __main__.<lambda>()>,
            {'Republican': defaultdict(int,
                         {'Republican': 0, 'Democratic': 4278}),
             'Democratic': defaultdict(int,
                         {'Republican': 0, 'Democratic': 5724})})
```

Reflections

The Naive Bayes model defaults to labeling virtually all tweets as Democratic in the sample, every Republican tweet was misclassified because the policy-focused vocabulary learned from convention speeches doesn’t transfer to the informal, hashtag-driven language of tweets. With features drawn from speeches, there’s almost no signal that a tweet is Republican, so the classifier collapses to one class. This highlights the domain mismatch and how relying solely on convention speech tokens fails when analyzing everyday tweets.