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. You can download the required DB from the shared dropbox or from blackboard

In [ ]:
```
import sqlite3
import nltk
import random
from string import punctuation
from nltk.classify import NaiveBayesClassifier
from nltk import FreqDist


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

In [ ]:
```
convention_db = sqlite3.connect("2020_conventions.db")
convention_cur = convention_db.cursor()
# List all tables
print(convention_cur.execute("SELECT name FROM sqlite_master WHERE type='table
```

```
[('conventions',)]
```

In [ ]:
```
print(convention_cur.execute("PRAGMA table_info(conventions);").fetchall())
```

```
[(0, 'district', 'TEXT', 0, None, 0), (1, 'candidate', 'TEXT', 0, None, 0),
(2, 'pull_time', 'DATETIME', 0, None, 0), (3, 'tweet_time', 'DATETIME', 0, Non
e, 0), (4, 'handle', 'TEXT', 0, None, 0), (5, 'is_retweet', 'INTEGER', 0, Non
e, 0), (6, 'tweet_id', 'TEXT', 0, None, 0), (7, 'tweet_text', 'TEXT', 0, None,
0), (8, 'likes', 'INTEGER', 0, None, 0), (9, 'replies', 'INTEGER', 0, None,
0), (10, 'retweets', 'INTEGER', 0, None, 0), (11, 'tweet_ratio', 'REAL', 0, No
ne, 0)]
```

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" exercise. First, we'll pull in the text for each party and prepare it for use in Naive Bayes.

In [ ]:
```
convention_data = []

query_results = convention_cur.execute(
    '''
    SELECT text, party
    FROM conventions
    WHERE party IN ('Republican', 'Democratic');
    '''
)

for row in query_results:
    text, party = row
    text = text.decode('utf-8') if isinstance(text, bytes) else text
    convention_data.append([text, party])
```

In [ ]:
```
# it's a best practice to close up your DB connection when you're done
convention_db.close()
```

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

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

Out[ ]:
```
[['Joe knows the world and the world knows him. He knows that our true strengt
h comes from setting an example that the world wants to follow, a nation that
stands with democracy, not dictators. A nation that can inspire and mobilize o
thers to overcome threats like climate change and terrorism, poverty, and dise
ase. But more than anything, what I know about Joe, what I know about Kamala,
is that they actually care about every American and that they care deeply abou
t this democracy. They believe that in a democracy, the right to vote is sacre
d and we should be making it easier for people to cast their ballots, not hard
er.',
  'Democratic'],
 ['It’s our honor to be in the company of great, great champions?',
  'Republican'],
 ['Joe Biden is that kind of leader. I created the Joyful Heart Foundation to
help survivors heal and to change the way our society response to sexual viole
nce. The vice president has worked tirelessly by our side to end the backlog o
f hundreds of thousands of untested rape kits. And our work will continue beca
use testing kits not only makes our country safer, but it sends a vital messag
e to survivors that what happened to them matters.',
  'Democratic'],
 ['Mike earned the trust of the people of his State and became the 50th Govern
or of Indiana. He delivered the largest state tax cut in Indiana history, expa
nded school choice, led the country in manufacturing, and helped more Hoosiers
get to work than ever before, but he wasn’t through.',
  'Republican'],
 ['I relive that horror in my mind every single day. My hope is that having yo
u relive it with me now, will help shake this country from this nightmare we’r
e witnessing in our cities and bring about positive, peaceful change. How did
we get to this point, where so many young people are callous and indifferent t
owards human life? This isn’t a video game, where you can commit mayhem and th
en just hit reset and bring all the characters back to life.',
  'Republican']]
```

It'll be useful for us to have a large sample size than 2020 affords, since those speeches tend to be long and contiguous. Let's make a new list-of-lists called `conv_sent_data`. Instead of each first entry in the sublists being an entire speech, make each first entry just a sentence from the speech. Feel free to use NLTK's `sent_tokenize` function.

In [ ]:
```
nltk.download('punkt')

conv_sent_data = []

for text, party in convention_data:
    sentences = nltk.sent_tokenize(text)
    for sentence in sentences:
        conv_sent_data.append([sentence, party])
```

```
random.choices(conv_sent_data, k=5)
```

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

Out[ ]:
```
[['One day, I was cleaning golf clubs when a man pulled into the parking lo
t.',
  'Republican'],
 ['John Lewis had the respect of everybody because he was the one who demonstr
ated the most courage.',
  'Democratic'],
 ['The Obama-Biden administration secretly launched a surveillance operation o
n the Trump campaign and silenced the many brave intelligence officials who sp
oke up against it.',
  'Republican'],
 ['My answer, going high is the only thing that works because when we go low,
when we use those same tactics of degrading and dehumanizing others, we just b
ecome part of the ugly noise that’s drowning out everything else.',
  'Democratic'],
 ['I stand here tonight, calling on all Americans to join us.', 'Republican']]
```

Again, let's look at some random entries.

In [ ]:
```
random.choices(conv_sent_data,k=5)
```

Out[ ]:
```
[['I took on the biggest banks and help take down one of the biggest for- prof
it colleges.',
  'Democratic'],
 ['He’s a Trojan horse with Bernie, AOC, Pelosi, Black Lives Matter, and his p
arty’s entire left wing, just waiting to execute their pro-criminal, anti-poli
ce, socialist policies.',
  'Republican'],
 ['Trump’s pledge to the American workers definitely means a lot to me because
I wouldn’t be where I’m at today.',
  'Republican'],
 ['The Biden-Harris vision for America leaves no room for people of faith.',
  'Republican'],
 ['I pray God’s blessings on you and your family and may God bless America.',
  'Republican']]
```

Now it's time for our final cleaning before modeling. Go through `conv_sent_data` and take the following steps:

1. Tokenize on whitespace
2. Remove punctuation
3. Remove tokens that fail the `isalpha` test
4. Remove stopwords
5. Casefold to lowercase
6. Join the remaining tokens into a string

In [ ]:
```
clean_conv_sent_data = []

for idx, sent_party in enumerate(conv_sent_data):
    sentence, party = sent_party
    tokens = sentence.split()
    tokens = [token.strip(punctuation) for token in tokens if token.strip(punc
    tokens = [token.lower() for token in tokens if token.lower() not in stopwo
    cleaned_sentence = ' '.join(tokens)
    clean_conv_sent_data.append((cleaned_sentence, party))

random.choices(clean_conv_sent_data, k=5)
```

Out[ ]:
```
[('thank', 'Democratic'),
 ('waiting washington act climate change', 'Democratic'),
 ('millions working families wondering feed kids worried evicted homes',
  'Democratic'),
 ('yeah', 'Democratic'),
 ('thank senator', 'Democratic')]
```

If that looks good, let's make our function to turn these into features. First we need to build our list of candidate words. I started my exploration at a cutoff of 5.

In [ ]:
```
word_cutoff = 5

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

word_dist = 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 fe
```

```
With a word cutoff of 5, we have 2239 as features in the model.
```

In [ ]:
```
def conv_features(text, fw):
    ret_dict = dict()
    tokens = text.split()
    for token in tokens:
        if token in fw:
            ret_dict[token] = True
    return(ret_dict)
```

In [ ]:
```
assert(len(feature_words) > 0)
assert(conv_features("obama was the president", feature_words) ==
       {'obama': True, 'president': True})
assert(conv_features("some people in america are citizens", feature_words) ==
       {'people': True, 'america': True, "citizens": 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 [ ]:
```
featuresets = [(conv_features(text,feature_words), party) for (text, party) in
```

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

test_size = 500
```

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

```
0.498
```

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

```
Most Informative Features
            reproductive = True           Democr : Republ =    229.9 : 1.0
                  comité = True           Republ : Democr =    143.9 : 1.0
              tecnología = True           Republ : Democr =    142.0 : 1.0
               postcards = True           Democr : Republ =     86.6 : 1.0
                 leftist = True           Republ : Democr =     75.0 : 1.0
           reinstituting = True           Republ : Democr =     73.1 : 1.0
                graduada = True           Republ : Democr =     71.2 : 1.0
           transparencia = True           Republ : Democr =     71.2 : 1.0
               polluters = True           Democr : Republ =     65.2 : 1.0
             duplicative = True           Republ : Democr =     58.0 : 1.0
                illegals = True           Republ : Democr =     50.5 : 1.0
              canvassers = True           Democr : Republ =     50.0 : 1.0
                 canvass = True           Democr : Republ =     49.2 : 1.0
          representantes = True           Republ : Democr =     48.0 : 1.0
              wealthiest = True           Democr : Republ =     41.4 : 1.0
                votantes = True           Republ : Democr =     39.2 : 1.0
                leftists = True           Republ : Democr =     37.3 : 1.0
                 livable = True           Democr : Republ =     35.8 : 1.0
               privatize = True           Democr : Republ =     35.4 : 1.0
                 richest = True           Democr : Republ =     32.8 : 1.0
                informes = True           Republ : Democr =     31.3 : 1.0
                 cruelly = True           Democr : Republ =     30.9 : 1.0
                  aliens = True           Republ : Democr =     30.0 : 1.0
             streamlines = True           Republ : Democr =     29.7 : 1.0
                 amnesty = True           Republ : Democr =     29.1 : 1.0
```

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

My naive bayes classifier achieved an accuracy of 48%, which is worse than random guessing but has lots of room for improvement in distinguishing Democratic and Republican tweets from 2020. The most informative features reveal clear ideological divides: Democratic Tweets: Emphasize social justice (reproductive, livable, cruelly), environmental concerns (polluters), and economic equity (wealthiest, richest). Terms like canvass and postcards highlight grassroots efforts, likely adapted for the COVID-19 context (e.g., mail-based voter outreach).

Republican Tweets: Focus on immigration (illegals, aliens, amnesty), anti-leftist rhetoric (leftist, leftists), and government efficiency (duplicative, streamlines). The presence of Spanish words (comité, tecnología, votantes) suggests targeted outreach to Hispanic voters, a key strategy in 2020.

The low accuracy is primarily due to a preprocessing mismatch: feature_words were built from cleaned data (lowercase, no punctuation, no stopwords), but convention_data contains raw tweets, leading to missed feature matches for instance healthcare and Healthcare may not be matched. Additionally, noise in the data (e.g., URLs, mentions, and terms like “rt”) and a large feature set (1246 features) may contribute to sparsity and overfitting. The Spanish-language features are interesting but might overfit to a small subset of tweets since the dataset is mostly English.

Note: I had originally made the mistake of using the congressional_data.db for training and had a 62% accuracy before realizing my mistake. Once that mistake was made I swapped the variables to match the tables from the correct db and my accuracy dropped by 14%, I wonder why if fit better with the other database, Anywho, just found that interesting.

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 [ ]:
```
cong_db = sqlite3.connect("congressional_data.db")
cong_cur = cong_db.cursor()
```

In [ ]:
```
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 consuming
```

In [ ]:
```
congressional_data = []

for candidate, party, tweet_text in results:
    tweet_text = tweet_text.decode('utf-8') if isinstance(tweet_text, bytes) e
    congressional_data.append([tweet_text, party])

clean_congressional_data = []

for text, party in congressional_data:
    tokens = text.split()
    tokens = [token.strip(punctuation) for token in tokens if token.strip(punc
    tokens = [token.lower() for token in tokens if token.lower() not in stopwo
    cleaned_text = ' '.join(tokens)
    clean_congressional_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 [ ]:
```
random.seed(20201014)

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

In [ ]:
```
for tweet, party in tweet_data_sample:
    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 classifier says {estimated_party}.'
    print("")
```

```
Here's our (cleaned) tweet: earlier today spoke house floor abt protecting hea
lth care women praised ppmarmonte work central coast
Actual party is Democratic and our classifier says Republican.

Here's our (cleaned) tweet: go tribe rallytogether
Actual party is Democratic and our classifier says Democratic.

Here's our (cleaned) tweet: apparently trump thinks easy students overwhelmed
crushing burden debt pay student loans trumpbudget
Actual party is Democratic and our classifier says Republican.

Here's our (cleaned) tweet: grateful first responders rescue personnel firefig
hters police volunteers working tirelessly keep people safe provide help putti
ng lives line
Actual party is Republican and our classifier says Republican.

Here's our (cleaned) tweet: make even greater kag
Actual party is Republican and our classifier says Republican.

Here's our (cleaned) tweet: cavs tie series repbarbaralee scared roadtovictory
Actual party is Democratic and our classifier says Democratic.

Here's our (cleaned) tweet: congrats belliottsd new gig sd city hall glad cont
inue
Actual party is Democratic and our classifier says Republican.

Here's our (cleaned) tweet: really close raised toward match right whoot major
s room help us get
Actual party is Democratic and our classifier says Republican.

Here's our (cleaned) tweet: today comment period plan expand offshore drilling
opened public days march share oppose proposed program directly trump administ
ration comments made email mail
Actual party is Democratic and our classifier says Republican.

Here's our (cleaned) tweet: celebrated years eastside commitment amp saluted c
ommunity leaders last awards dinner
Actual party is Democratic and our classifier says Republican.
```

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

In [ ]:
```
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(clean_congressional_data)

for idx, tp in enumerate(clean_congressional_data):
    tweet, party = tp
    features = conv_features(tweet, feature_words)
    estimated_party = classifier.classify(features)

    results[party][estimated_party] += 1

    if idx > num_to_score:
        break
```

In [ ]:
```
# Print the results
print("Confusion Matrix (Actual vs. Estimated):")
print("True \\ Predicted | Democratic | Republican")
print(f"Democratic       | {results['Democratic']['Democratic']}        | {resu
print(f"Republican       | {results['Republican']['Democratic']}        | {resu

correct = results['Democratic']['Democratic'] + results['Republican']['Republi
total = sum(sum(results[p].values()) for p in parties)
accuracy = correct / total if total > 0 else 0
print(f"\nAccuracy on {total} scored tweets: {accuracy:.2f}")

cong_db.close()
```

```
Confusion Matrix (Actual vs. Estimated):
True \ Predicted | Democratic | Republican
Democratic       | 998        | 4726
Republican       | 694        | 3584

Accuracy on 10002 scored tweets: 0.46
```

Reflections

The classifier, trained on 2020 convention speeches, scored a 46% accuracy on 10,002 congressional tweets from 2018, as shown in the confusion matrix. It correctly classified 998 of 5,724 Democratic tweets but misclassified 4,726 as Republican, indicating a strong bias toward the Republican label. For Republican tweets, it correctly identified 3,584 of 4,278, performing better at 84% precision, though 694 were mislabeled as Democratic. The low accuracy, below a 50% random baseline, stems from the mismatch between formal speeches and informal tweets, compounded by the temporal gap between 2020 and 2018 political contexts. The classifier likely overfits to Republican speech patterns, such as deregulation themes, while struggling with overlapping Democratic terms. This highlights the difficulty of cross-domain classification and suggests retraining with a mixed dataset or refining features for tweets. Would you like to adjust the feature set or retrain with different data?