Political Naive Bayes

June 1, 2022

0.1 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.

[1]:
```
import sqlite3
import nltk
import random
import numpy as np
from collections import Counter, defaultdict

# Feel free to include your text patterns functions
# from text_functions_solutions import clean_tokenize, get_patterns
from nltk.corpus import stopwords
import re
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
sw = set(stopwords.words('english'))
```

[3]:
```
convention_db = sqlite3.connect("2020_Conventions.db")
convention_cur = convention_db.cursor()
```

0.1.1 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.

[4]:
```
def clean_text(text, is_tweet=False):

    text = text.lower()

    if is_tweet:
        # Keep the hashtag and tagging so it's easier to drop later
        text = re.sub(r"[^a-zA-Z0-9 #@]", '', text)
    else:
        # Only keep alpha-numeric values + space
        text = re.sub(r"[^a-zA-Z0-9 ]", '', text)
```

```
    # 2 or more space into one
    text = re.sub('\s+', ' ', text)

    # tokenize
    text = text.split(' ')

    # remove stopwords from tokens
    text = [x for x in text if x not in sw]

    if is_tweet:
        # each tweet contains url/tagging at the end of the tweet
        # drop them if so
        text = [x for x in text if 'http' not in x and '@' not in x]

    return ' '.join(text)
```

[5]:
```
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 * from conventions
                            ''')


cols = [x[0] for x in query_results.description]

print('Columns are {}'.format(cols))

for row in query_results:

    text = row[-3]
    text = clean_text(text)

    convention_data.append([text, row[0]])

print('Total length of {}'.format(len(convention_data)))
```

```
Columns are ['party', 'night', 'speaker', 'speaker_count', 'time', 'text',
'text_len', 'file']
Total length of 2541
```

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

[6]:
```
random.choices(convention_data,k=10)
```

[6]:
```
[['donald trump helped make happen years later opponents running president know
turned im proud job donald trump done president dont always agree occasional
policy differences far outweighed significant agreements important simple
agreement accomplishment president trump gets things done congress wanted
socalled revenueneutral tax plan donald trump worked together make sure true tax
cut brought president trump idea better less expensive health insurance called
association health plans donald trump overturned years red tape bureaucrats made
happen others talked criminal justice reform president trump actually signed
first step act first real reform generation one sought undo harm others like joe
biden done us lament biden crime bill locked generation young black men remember
biden bragged bill still wreaks havoc among people color',
  'Republican'],
 ['racist coward speaker 92 013749 call cops speaker 93 013749 going kill
speaker 94 013749 im rape speaker 95 013758 mark mccloskey says family
threatened violence',
  'Republican'],
 ['four years imagine achieve simply working president believe presidents vision
future stand tonight calling americans join us doesnt matter look like love
worship gender job youre traditional democrat whos become disillusioned radical
party become stand us welcome america needs patriots rush defense fellow
americans promise tent free free speak truth choose journey define life power go
far aim aim higher keep going americans idealists dreamers lovers adventure
rugged independent dont make excuses make impossible reality',
  'Republican'],
 ['dont even know cover payroll isnt going well scary', 'Democratic'],
 ['beau diagnosed brain cancer nobody knew going secret service theyre supposed
react life youre would whisper im praying hunter biden 020910 strength holds
family together know make us whole dr',
  'Democratic'],
 ['weve always willing thought guy right cross aisle lock arms good country',
  'Democratic'],
 ['youve pledged undying loyalty american people american constitution american
way life history heritage united states preserve pass next generation culture
traditions values uphold live rights dear every american granted us granted god
enshrined glorious bill rights support protect defend citizens youre stewards
magnificent nation family comprised every race color religion creed united bonds
love one people sharing one home saluting one great american flag
congratulations may god bless may god bless great country america',
  'Republican'],
 ['day came phone rang vice president telling loves cover whole article one best
issues arrive ever read one many reasons wanted tonight joe remind joe biden
knows read also reads everything policy expert certainly dont pretend one gut
feeling fairness whats right excited little going hear joe biden plans america
plans strong economy helps working people small businesses billionaires plans
raise wages nurses teachers grocery workers getting us crisis god love',
```

```
 'Democratic'],
['correct', 'Democratic'],
['second lady teaching full time eight years 15 credits semester dr',
 'Democratic']]
```

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.

[7]:
```
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␣
↪features in the model.")
```

```
With a word cutoff of 5, we have 2378 as features in the model.
```

[8]:
```
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','jumps'},
            then this would return a dictionary of
            {'quick' : True,
             'fox' :    True}

    """
    ret_dict = {token: True for token in set(text.split(' ')) if token in fw}
```

```
    return(ret_dict)
```

[9]:
```
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.

[10]:
```
featuresets = [(conv_features(text,feature_words), party) for (text, party) in␣
↪convention_data]
```

[11]:
```
random.seed(20220507)
random.shuffle(featuresets)

test_size = 500
```

[12]:
```
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
```

[13]:
```
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
                  beliefs = True           Republ : Democr =     13.0 : 1.0
                countries = True           Republ : Democr =     13.0 : 1.0
                  defense = True           Republ : Democr =     13.0 : 1.0
                   defund = 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
```

```
                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
```

[14]:
```
cd = pd.DataFrame(convention_data, columns=['Text', 'Party'])
cd.head()
```

[14]:

	Text	Party
0	skip content company careers press freelancers…	Democratic
1	im calling full session 48th quadrennial natio…	Democratic
2	every four years come together reaffirm democr…	Democratic
3	fight perfect union fighting soul country live…	Democratic
4	must come together defeat donald trump elect j…	Democratic

[15]:
```
cd['Party'].value_counts()
```

[15]:
```
Democratic    1551
Republican     990
Name: Party, dtype: int64
```

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

0.1.2 My Observations

I tried a few different word_cutoff sizes and seems that 5 is working the best for this task. One main thing looking odd from the list is how the classifier model is favoring much on Republican.

Converting the `convention_data` into a Pandas dataframe, we can see the counts of each party is quite not balanced. Even though the number of `Replublican` is lower than the other, the model decided to favor more on it. Not only that, the top 25 features (or words) that are appearing above do not particularly favor one party to another (or it should not). Like the word "media" for example, in a normal sense it is a word that should quite be neutral between both parties but the classifier decided to use this word as one top feature to distinguish the label. This may be due to it's only taking account the number of a word appearing across the texts and not the actual meaning (semantics) behind the word. I'm assuming that the word 'china' is appearing most in one party (based on the print above) and not any much in the other. Because of this, its accuracy may not be so high, in fact lower than 50%.

Another possible solution would be configuring some of words appearing in stopwords like your or my because "your freedom got destroyed" vs "my freedom got destryoed" have different meanings but if those stopwords are dropped, they will end up "freedom got destroyed" having the same meaning. Maybe this could help distinguishing the label?

0.2 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.

[16]:
```
cong_db = sqlite3.connect("congressional_data.db")
cong_cur = cong_db.cursor()
```

[26]:
```
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
```

[26]:
```
# Now fill up tweet_data with sublists like we did on the convention speeches.
# Note that this may take a bit of time, since we have a lot of tweets.

# Decode the tweets since it is in byte
tweet_data = [[clean_text(row[2].decode(), is_tweet=True), row[1]] for row in␣
 ↪results]
```

```
ERROR:root:Internal Python error in the inspect module.
Below is the traceback from this internal error.

ERROR:root:Internal Python error in the inspect module.
Below is the traceback from this internal error.


Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 3441, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/1172985013.py",
line 5, in <module>
    tweet_data = [[clean_text(row[2].decode(), is_tweet=True), row[1]] for row
in results]
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/1172985013.py",
line 5, in <listcomp>
    tweet_data = [[clean_text(row[2].decode(), is_tweet=True), row[1]] for row
```

```
in results]
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/700154141.py",
line 24, in clean_text
    text = [x for x in text if 'http' not in x and '@' not in x]
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/700154141.py",
line 24, in <listcomp>
    text = [x for x in text if 'http' not in x and '@' not in x]
KeyboardInterrupt

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 2061, in showtraceback
    stb = value._render_traceback_()
AttributeError: 'KeyboardInterrupt' object has no attribute '_render_traceback_'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 1101, in get_records
    return _fixed_getinnerframes(etb, number_of_lines_of_context, tb_offset)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 248, in wrapped
    return f(*args, **kwargs)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 281, in _fixed_getinnerframes
    records = fix_frame_records_filenames(inspect.getinnerframes(etb, context))
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 1541,
in getinnerframes
    frameinfo = (tb.tb_frame,) + getframeinfo(tb, context)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 1499,
in getframeinfo
    filename = getsourcefile(frame) or getfile(frame)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 709, in
getsourcefile
    if getattr(getmodule(object, filename), '__loader__', None) is not None:
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 752, in
getmodule
    f = getabsfile(module)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 721, in
getabsfile
    _filename = getsourcefile(object) or getfile(object)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 706, in
getsourcefile
```

```
    if os.path.exists(filename):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/genericpath.py", line 19,
in exists
    os.stat(path)
KeyboardInterrupt
Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 3441, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/1172985013.py",
line 5, in <module>
    tweet_data = [[clean_text(row[2].decode(), is_tweet=True), row[1]] for row
in results]
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/1172985013.py",
line 5, in <listcomp>
    tweet_data = [[clean_text(row[2].decode(), is_tweet=True), row[1]] for row
in results]
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/700154141.py",
line 24, in clean_text
    text = [x for x in text if 'http' not in x and '@' not in x]
  File
"/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/700154141.py",
line 24, in <listcomp>
    text = [x for x in text if 'http' not in x and '@' not in x]
KeyboardInterrupt

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 2061, in showtraceback
    stb = value._render_traceback_()
AttributeError: 'KeyboardInterrupt' object has no attribute '_render_traceback_'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 3361, in run_ast_nodes
    if (await self.run_code(code, result,  async_=asy)):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 3458, in run_code
    self.showtraceback(running_compiled_code=True)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 2063, in showtraceback
```

```
    stb = self.InteractiveTB.structured_traceback(etype,
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 1367, in structured_traceback
    return FormattedTB.structured_traceback(
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 1267, in structured_traceback
    return VerboseTB.structured_traceback(
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 1124, in structured_traceback
    formatted_exception = self.format_exception_as_a_whole(etype, evalue, etb,
number_of_lines_of_context,
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 1082, in format_exception_as_a_whole
    last_unique, recursion_repeat = find_recursion(orig_etype, evalue, records)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 382, in find_recursion
    return len(records), 0
TypeError: object of type 'NoneType' has no len()

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/interactiveshell.py", line 2061, in showtraceback
    stb = value._render_traceback_()
AttributeError: 'TypeError' object has no attribute '_render_traceback_'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 1101, in get_records
    return _fixed_getinnerframes(etb, number_of_lines_of_context, tb_offset)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 248, in wrapped
    return f(*args, **kwargs)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/site-
packages/IPython/core/ultratb.py", line 281, in _fixed_getinnerframes
    records = fix_frame_records_filenames(inspect.getinnerframes(etb, context))
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 1541,
in getinnerframes
    frameinfo = (tb.tb_frame,) + getframeinfo(tb, context)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 1499,
in getframeinfo
    filename = getsourcefile(frame) or getfile(frame)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 709, in
getsourcefile
    if getattr(getmodule(object, filename), '__loader__', None) is not None:
```

```
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/inspect.py", line 755, in
getmodule
    os.path.realpath(f)] = module.__name__
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/posixpath.py", line 391,
in realpath
    path, ok = _joinrealpath(filename[:0], filename, {})
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/posixpath.py", line 424,
in _joinrealpath
    newpath = join(path, name)
  File "/Users/han/opt/anaconda3/envs/py/lib/python3.9/posixpath.py", line 83,
in join
    if b.startswith(sep):
KeyboardInterrupt
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
    [… skipping hidden 1 frame]

/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/1172985013.py i␣
↪<module>
      4 # Decode the tweets since it is in byte
----> 5 tweet_data = [[clean_text(row[2].decode(), is_tweet=True), row[1]] for␣
↪row in results]

/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/1172985013.py i␣
↪<listcomp>(.0)
      4 # Decode the tweets since it is in byte
----> 5 tweet_data = [[clean_text(row[2].decode(), is_tweet=True), row[1]] for␣
↪row in results]

/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/700154141.py in␣
↪clean_text(text, is_tweet)
     23         # drop them if so
---> 24         text = [x for x in text if 'http' not in x and '@' not in x]
     25

/var/folders/cp/kcsvfqz95qj_s93tp2sqb_f80000gn/T/ipykernel_4547/700154141.py in␣
↪<listcomp>(.0)
     23         # drop them if so
---> 24         text = [x for x in text if 'http' not in x and '@' not in x]
     25

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
```

```
~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/
 ↪interactiveshell.py in showtraceback(self, exc_tuple, filename, tb_offset,␣
 ↪exception_only, running_compiled_code)
   2060                         # in the engines. This should return a list of␣
 ↪strings.
-> 2061                         stb = value._render_traceback_()
   2062                     except Exception:

AttributeError: 'KeyboardInterrupt' object has no attribute '_render_traceback_

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/
 ↪interactiveshell.py in run_ast_nodes(self, nodelist, cell_name, interactivity␣
 ↪compiler, result)
   3360                         asy = compare(code)
-> 3361                     if (await self.run_code(code, result,  async_=asy)):
   3362                         return True

    [… skipping hidden 1 frame]

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/
 ↪interactiveshell.py in showtraceback(self, exc_tuple, filename, tb_offset,␣
 ↪exception_only, running_compiled_code)
   2062                     except Exception:
-> 2063                         stb = self.InteractiveTB.
 ↪structured_traceback(etype,
   2064                                               value, tb,␣
 ↪tb_offset=tb_offset)

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
 ↪structured_traceback(self, etype, value, tb, tb_offset,␣
 ↪number_of_lines_of_context)
   1366             self.tb = tb
-> 1367         return FormattedTB.structured_traceback(
   1368             self, etype, value, tb, tb_offset,␣
 ↪number_of_lines_of_context)

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
 ↪structured_traceback(self, etype, value, tb, tb_offset,␣
 ↪number_of_lines_of_context)
   1266             # Verbose modes need a full traceback
-> 1267             return VerboseTB.structured_traceback(
   1268                 self, etype, value, tb, tb_offset,␣
 ↪number_of_lines_of_context
```

```
~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪structured_traceback(self, etype, evalue, etb, tb_offset,␣
↪number_of_lines_of_context)
   1123
-> 1124         formatted_exception = self.format_exception_as_a_whole(etype,␣
↪evalue, etb, number_of_lines_of_context,
   1125                                                                tb_offse )

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪format_exception_as_a_whole(self, etype, evalue, etb,␣
↪number_of_lines_of_context, tb_offset)
   1081
-> 1082         last_unique, recursion_repeat = find_recursion(orig_etype,␣
↪evalue, records)
   1083

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪find_recursion(etype, value, records)
    381     if not is_recursion_error(etype, value, records):
--> 382         return len(records), 0
    383

TypeError: object of type 'NoneType' has no len()

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/
↪interactiveshell.py in showtraceback(self, exc_tuple, filename, tb_offset,␣
↪exception_only, running_compiled_code)
   2060                         # in the engines. This should return a list of␣
↪strings.
-> 2061                         stb = value._render_traceback_()
   2062                     except Exception:

AttributeError: 'TypeError' object has no attribute '_render_traceback_'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/async_helpers.
↪py in _pseudo_sync_runner(coro)
     66     """
     67     try:
---> 68         coro.send(None)
     69     except StopIteration as exc:
     70         return exc.value
```

```
~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/
↪interactiveshell.py in run_cell_async(self, raw_cell, store_history, silent,␣
↪shell_futures, transformed_cell, preprocessing_exc_tuple)
   3167                     interactivity = 'async'
   3168
-> 3169                 has_raised = await self.run_ast_nodes(code_ast.body,␣
↪cell_name,
   3170                        interactivity=interactivity, compiler=compiler,␣
↪result=result)
   3171

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/
↪interactiveshell.py in run_ast_nodes(self, nodelist, cell_name, interactivity␣
↪compiler, result)
   3378             if result:
   3379                 result.error_before_exec = sys.exc_info()[1]
-> 3380             self.showtraceback()
   3381             return True
   3382

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/
↪interactiveshell.py in showtraceback(self, exc_tuple, filename, tb_offset,␣
↪exception_only, running_compiled_code)
   2061                         stb = value._render_traceback_()
   2062                     except Exception:
-> 2063                         stb = self.InteractiveTB.
↪structured_traceback(etype,
   2064                                               value, tb,␣
↪tb_offset=tb_offset)
   2065

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪structured_traceback(self, etype, value, tb, tb_offset,␣
↪number_of_lines_of_context)
   1365         else:
   1366             self.tb = tb
-> 1367         return FormattedTB.structured_traceback(
   1368             self, etype, value, tb, tb_offset,␣
↪number_of_lines_of_context)
   1369

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪structured_traceback(self, etype, value, tb, tb_offset,␣
↪number_of_lines_of_context)
   1265         if mode in self.verbose_modes:
   1266             # Verbose modes need a full traceback
-> 1267             return VerboseTB.structured_traceback(
   1268                 self, etype, value, tb, tb_offset,␣
↪number_of_lines_of_context
```

```
   1269             )

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪structured_traceback(self, etype, evalue, etb, tb_offset,␣
↪number_of_lines_of_context)
   1140         chained_exc_ids = set()
   1141         while evalue:
-> 1142             formatted_exceptions += self.
↪format_exception_as_a_whole(etype, evalue, etb, lines_of_context,
   1143                                                                  ␣
↪chained_exceptions_tb_offset)
   1144             exception = self.get_parts_of_chained_exception(evalue)

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪format_exception_as_a_whole(self, etype, evalue, etb,␣
↪number_of_lines_of_context, tb_offset)
   1080
   1081
-> 1082         last_unique, recursion_repeat = find_recursion(orig_etype,␣
↪evalue, records)
   1083
   1084         frames = self.format_records(records, last_unique,␣
↪recursion_repeat)

~/opt/anaconda3/envs/py/lib/python3.9/site-packages/IPython/core/ultratb.py in␣
↪find_recursion(etype, value, records)
    380     # first frame (from in to out) that looks different.
    381     if not is_recursion_error(etype, value, records):
--> 382         return len(records), 0
    383
    384     # Select filename, lineno, func_name to track frames with

TypeError: object of type 'NoneType' has no len()
```

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...

[19]:
```
random.seed(20201014)

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

[20]:
```
featuresets = [(conv_features(text,feature_words), party) for (text, party) in␣
↪convention_data]
```

[21]:
```
for tweet, party in tweet_data_sample:
    fw = conv_features(tweet, feature_words)
    estimated_party = classifier.classify(fw)
```

```
    # Fill in the right-hand side above with code that estimates the actual␣
↪party

    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: earlier today spoke house floor abt protecting
health care women praised work central coast
Actual party is Democratic and our classifer says Republican.

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

Here's our (cleaned) tweet: grateful first responders rescue personnel
firefighters police volunteers working tirelessly keep people safe provide
muchneeded help putting lives
Actual party is Republican and our classifer says Republican.

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

Here's our (cleaned) tweet: 1hr tie series 22 im #allin216 scared #roadtovictory
Actual party is Democratic and our classifer says Republican.

Here's our (cleaned) tweet: congrats new gig sd city hall glad continue serve
Actual party is Democratic and our classifer says Republican.

Here's our (cleaned) tweet: really close 3500 raised toward match right whoot
thats 7000 nonmath majors room help us get
Actual party is Democratic and our classifer says Republican.

Here's our (cleaned) tweet: today comment period plan expand offshore drilling
opened public 60 days march 9 share oppose proposed program directly trump
administration comments made email mail
Actual party is Democratic and our classifer says Republican.

Here's our (cleaned) tweet: celebrated 22 years eastside commitment amp saluted
community leaders last nights awards dinner
Actual party is Democratic and our classifer says Republican.
```

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

[22]:
```
# dictionary of counts by actual party and estimated party.
# first key is actual, second is estimated
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)

y_pred = []
y_true = []

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.
    fw = conv_features(tweet, feature_words)

    # get the estimated party
    estimated_party = classifier.classify(fw)

    results[party][estimated_party] += 1

    y_true.append(party)
    y_pred.append(estimated_party)

    if idx > num_to_score :
        break

y_true = pd.Series(y_true, name='Actual')
y_pred = pd.Series(y_pred, name='Pred')
```

[23]:
```
results
```

[23]:
```
defaultdict(<function __main__.<lambda>()>,
            {'Republican': defaultdict(int,
                        {'Republican': 3714, 'Democratic': 564}),
             'Democratic': defaultdict(int,
                        {'Republican': 4829, 'Democratic': 895})})
```

[24]:
```
pd.crosstab(y_true, y_pred)
```

[24]:

Pred	Democratic	Republican
Actual		
Democratic	895	4829
Republican	564	3714

0.2.1 Reflections

As mentioned in the first part, it seems that the classifier is favoring the `Repulican` party than the other which can easily be seen by the results or the confusion matrix right above. This could be due to imbalanced dataset or selection of keywords (shown above with 25 top features). To better or improve the performance of the model, choosing which set of words to use to distinguish between different labels is important. At this moment, regardless of the actual semantics of texts, it's naively making most of its prediction as one (`Republican`) hoping to achieve the best result.

A few ways to improve the model's accuracy exist as mentioned earlier in the part 1. Another possible way would be using different model (of course). Because there is no hyperparameter tuning for this model, there is limit to what we can do to improve the model. Also actually using the hashtags could improve as well but since there are a very few data we can utilize (not even 10k), this may not be viable.