Lab3-assignment-sentiment

March 3, 2025

1 Lab3 - Assignment Sentiment

This notebook describes the LAB-3 assignment of the Text Mining course. It is about sentiment analysis.

The aims of the assignment are: * Learn how to run a rule-based sentiment analysis module (VADER) * Learn how to run a machine learning sentiment analysis module (Scikit-Learn/ Naive Bayes) * Learn how to run scikit-learn metrics for the quantitative evaluation * Learn how to perform and interpret a quantitative evaluation of the outcomes of the tools (in terms of Precision, Recall, and F1) * Learn how to evaluate the results qualitatively (by examining the data) * Get insight into differences between the two applied methods * Get insight into the effects of using linguistic preprocessing * Be able to describe differences between the two methods in terms of their results * Get insight into issues when applying these methods across different domains

In this assignment, you are going to create your own gold standard set from 50 tweets. You will the VADER and scikit-learn classifiers to these tweets and evaluate the results by using evaluation metrics and inspecting the data.

We recommend you go through the notebooks in the following order: * **Read the assignment (see below)** * **Lab3.2-Sentiment-analysis-with-VADER.ipynb** * **Lab3.3-Sentiment-analysis.with-scikit-learn.ipynb** * **Answer the questions of the assignment (see below) using the provided notebooks and submit**

In this assignment you are asked to perform both quantitative evaluations and error analyses: * a quantitative evaluation concerns the scores (Precision, Recall, and F1) provided by scikit's classification_report. It includes the scores per category, as well as micro and macro averages. Discuss whether the scores are balanced or not between the different categories (positive, negative, neutral) and between precision and recall. Discuss the shortcomings (if any) of the classifier based on these scores * an error analysis regarding the misclassifications of the classifier. It involves going through the texts and trying to understand what has gone wrong. It servers to get insight in what could be done to improve the performance of the classifier. Do you observe patterns in misclassifications? Discuss why these errors are made and propose ways to solve them.

1.1 Credits

The notebooks in this block have been originally created by Marten Postma and Isa Maks. Adaptations were made by Filip Ilievski.

1.2 Part I: VADER assignments

1.2.1 Preparation (nothing to submit):

To be able to answer the VADER questions you need to know how the tool works. * Read more about the VADER tool in this blog.
* VADER provides 4 scores (positive, negative, neutral, compound). Be sure to understand what they mean and how they are calculated. * VADER uses rules to handle linguistic phenomena such as negation and intensification. Be sure to understand which rules are used, how they work, and why they are important. * VADER makes use of a sentiment lexicon. Have a look at the lexicon. Be sure to understand which information can be found there (lemma?, wordform?, part-of-speech?, polarity value?, word meaning?) What do all scores mean? https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/vader_lexicon.txt)

1.2.2 [3.5 points] Question1:

Regard the following sentences and their output as given by VADER. Regard sentences 1 to 7, and explain the outcome **for each sentence**. Take into account both the rules applied by VADER and the lexicon that is used. You will find that some of the results are reasonable, but others are not. Explain what is going wrong or not when correct and incorrect results are produced.

```
INPUT SENTENCE 1 I love apples
VADER OUTPUT {'neg': 0.0, 'neu': 0.192, 'pos': 0.808, 'compound': 0.6369}
```

- The words "I" and "apples" are neutral, while "love" is positive according to the VADER lexicon. At first glance it might seem odd that the positive score is higher than the neutral score, but this is reasonale since the word love has a very strong positive sentiment. The compound score of 0.6369 shows that VADER correctly identifies the sentence as positive overall.

```
INPUT SENTENCE 2 I don't love apples
VADER OUTPUT {'neg': 0.627, 'neu': 0.373, 'pos': 0.0, 'compound': -0.5216}
```

- Given that the compound score of -0,5216 is close to -1, VADER considers the sentence to have a negative sentiment overall. While one might expect a higher positive score due to the use of the word "love", the use of "don't" negates the positive sentiment of the word love. This explains the positive score of 0. Overall, the results are reasonable.

```
INPUT SENTENCE 3 I love apples :-)
VADER OUTPUT {'neg': 0.0, 'neu': 0.133, 'pos': 0.867, 'compound': 0.7579}
```

- The sentence "I love apples :-)" has a compound score of 0.7579, which indicates an overall positive sentiment. The smiley face emoticon is recognised by VADER as having a positive effect, so the positive score (0.867) is slightly higher than in the sentence "I love apples" (which had a positive score of 0.808). Overall, the results are reasonable, as the smiley face further increases the sentiment.

```
INPUT SENTENCE 4 These houses are ruins
VADER OUTPUT {'neg': 0.492, 'neu': 0.508, 'pos': 0.0, 'compound': -0.4404}
```

- The negative score of 0.492 comes from the word "ruins," which carries a negative sentiment. All of the other words are neutral and thus explain the neutral score of 0.508. Since there are no positive words in the sentence, the positive score is 0.0. The compound score of -0.4404

reflects the sentiment as slightly negative, but not strongly negative, due to the balance between the neutral and negative scores.

```
INPUT SENTENCE 5 These houses are certainly not considered ruins
VADER OUTPUT {'neg': 0.0, 'neu': 0.51, 'pos': 0.49, 'compound': 0.5867}
```

- VADER considers this sentence to have a slightly positive sentiment given that the compound score is about 0.5. The word "certainly" indicates a strong sentiment and the word "ruins", like in the sentence above, has a negative sentiment, but the use of the word "not" directly after it makes the sentiment positive. The positive and neutral socres are not that reasonable, because the use of the word certainly shows a stronger sentiment, so the positive value should have been a bit higher, making the neutral score lower.

```
INPUT SENTENCE 6 He lies in the chair in the garden
VADER OUTPUT {'neg': 0.286, 'neu': 0.714, 'pos': 0.0, 'compound': -0.4215}
```

- The negative score of 0.286 likely comes from the word "lies," which can have both a neutral meaning (i.e., lying down) and a negative one (i.e., telling lies). VADER could have interpreted it as negative, which is incorrect in this context. The rest of the sentence is neutral which explains the high neutral score (0.714).

```
INPUT SENTENCE 7 This house is like any house
VADER OUTPUT {'neg': 0.0, 'neu': 0.667, 'pos': 0.333, 'compound': 0.3612}
```

- The word "like" contributes to the positive score of 0.333 since it is associated with positive sentiment in the VADER lexicon. However, in this context the word "like" is used for comparison, so the positive score is incorrect. The neutral score of 0.667 is expected, as the sentence contains words like "This", "house" and "is" and does not have any lexicons with a strong sentiment.

1.2.3 [Points: 2.5] Exercise 2: Collecting 50 tweets for evaluation

Collect 50 tweets. Try to find tweets that are interesting for sentiment analysis, e.g., very positive, neutral, and negative tweets. These could be your own tweets (typed in) or collected from the Twitter stream. If you have trouble accessing Twitter, try to find an existing dataset (on websites like kaggle or huggingface).

We will store the tweets in the file **my_tweets.json** (use a text editor to edit). For each tweet, you should insert: * sentiment analysis label: negative | neutral | positive (this you determine yourself, this is not done by a computer) * the text of the tweet * the Tweet-URL

from:

```
    "1": {
        "sentiment_label": "",
        "text_of_tweet": "",
        "tweet_url": "",
```

to:

```
"1": {
        "sentiment_label": "positive",
        "text_of_tweet": "All across America people chose to get involved, get engaged and sta
```

```
        "tweet_url" : "https://twitter.com/BarackObama/status/946775615893655552",
    },
```

You can load your tweets with human annotation in the following way.

source = https://www.kaggle.com/datasets/ahmedshahriarsakib/tweet-sample

[1]:
```
import json
```

[2]:
```
my_tweets = json.load(open('my_tweets.json'))
```

[3]:
```
for id_, tweet_info in my_tweets.items():
    print(id_, tweet_info)
    break
```

```
1 {'sentiment_label': 'positive', 'text_of_tweet': 'All across America people
chose to get involved, get engaged and stand up. Each of us can make a
difference, and all of us ought to try. So go keep changing the world in 2018.',
'tweet_url': 'https://twitter.com/BarackObama/status/946775615893655552'}
```

1.2.4 [5 points] Question 3:

Run VADER on your own tweets (see function **run_vader** from notebook **Lab2-Sentiment-analysis-using-VADER.ipynb**). You can use the code snippet below this explanation as a starting point. [2.5 points] a. Perform a quantitative evaluation. Explain the different scores, and explain which scores are most relevant and why.

- The classification report shows the model's performance across three sentiment classes: negative, neutral, and positive. Precision indicates how accurate the model's predictions were for each class, while recall measures how well it identified all instances of each sentiment. A higher F1-score reflects a better overall performance. The model performs well for negative sentiment, with high precision (0.79) and recall (0.94), but struggles with neutral sentiment, showing a lower recall (0.44) and F1-score (0.56). The weighted average F1-score of 0.72 suggests the model performs well overall, but there is room for improvement, especially with neutral tweets.

[2.5 points] b. Perform an error analysis: select 10 positive, 10 negative and 10 neutral tweets that are not correctly classified and try to understand why. Refer to the VADER-rules and the VADER-lexicon. Of course, if there are less than 10 errors for a category, you only have to check those. For example, if there are only 5 errors for positive tweets, you just describe those.

- POSITIVE TWEETS MISCLASSIFIED:

1. Tweet: Nothing like a fresh cup of coffee in the morning to start the day right! | Predicted as: negative
2. Tweet: What an incredible concert! One for the books! | Predicted as: neutral
3. Tweet: Just booked my first solo trip! Can't wait to explore new places. | Predicted as: neutral

- NEGATIVE TWEETS MISCLASSIFIED:

1. Tweet: Can't believe how much the rent prices have gone up in this area. Unbelievable. | Predicted as: positive

- NEUTRAL TWEETS MISCLASSIFIED:

1. Tweet: Just finished watching the new episode. It was okay, not great but not terrible either. | Predicted as: positive
2. Tweet: The new phone update has some interesting changes. | Predicted as: positive
3. Tweet: The movie I watched last night was just okay, not as good as the reviews said. | Predicted as: negative
4. Tweet: The news today was a mix of interesting stories. | Predicted as: positive
5. Tweet: Just read an interesting article. Not sure how I feel about it yet. | Predicted as: positive
6. Tweet: The book I'm reading is slow to start, but I'm hoping it picks up soon. | Predicted as: positive
7. Tweet: I've been thinking about starting a new fitness routine. | Predicted as: positive
8. Tweet: Not sure how to feel about the news today. | Predicted as: negative
9. Tweet: Weather is looking mild for the weekend. No surprises expected. | Predicted as: negative

- The error analysis shows that the VADER sentiment analysis model often misclassifies tweets beacause it relies on certain keywords that may not always capture the full context of the tweet. Positive tweets can be misclassified as negative or neutral because VADER sometimes misses subtle positive expressions or misinterprets neutral phrases as negative. In negative tweets, VADER might mistake words with dual meanings, like "unbelievable," for positive sentiment. Neutral tweets are frequently misclassified as positive because the model focuses too much on positive words like "interesting." These misclassifications show how difficult it can be to detect sentiment, when nuance is involved.

[4]:
```python
def vader_output_to_label(vader_output):
    """
    map vader output e.g.,
    {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.4215}
    to one of the following values:
    a) positive float -> 'positive'
    b) 0.0 -> 'neutral'
    c) negative float -> 'negative'

    :param dict vader_output: output dict from vader

    :rtype: str
    :return: 'negative' | 'neutral' | 'positive'
    """
    compound = vader_output['compound']

    if compound < 0:
        return 'negative'
    elif compound == 0.0:
        return 'neutral'
    elif compound > 0.0:
        return 'positive'
```

```
assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound':␣
 ↪0.0}) == 'neutral'
assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound':␣
 ↪0.01}) == 'positive'
assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound':␣
 ↪-0.01}) == 'negative'
```

[5]:
```
from nltk.sentiment import vader
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.metrics import classification_report

tweets = []
all_vader_output = []
gold = []

# settings (to change for different experiments)
to_lemmatize = True
pos = set()
vader_model = SentimentIntensityAnalyzer()

for id_, tweet_info in my_tweets.items():
    the_tweet = tweet_info['text_of_tweet']
    vader_output = vader_model.polarity_scores(the_tweet) # run vader
    vader_label = vader_output_to_label(vader_output)# convert vader output to␣
 ↪category

    tweets.append(the_tweet)
    all_vader_output.append(vader_label)
    gold.append(tweet_info['sentiment_label'])

    # Print tweet and its classification
    print(f"Tweet: {the_tweet}")
    print(f"Predicted Sentiment: {vader_label}")
    print(f"Actual Sentiment: {tweet_info['sentiment_label']}")
    print()
```

```
Tweet: All across America people chose to get involved, get engaged and stand
up. Each of us can make a difference, and all of us ought to try. So go keep
changing the world in 2018.
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: This traffic is absolutely ridiculous. Been stuck for an hour and haven't
moved an inch.
Predicted Sentiment: negative
```

```
Predicted Sentiment: negative
Actual Sentiment: neutral

Tweet: Weather is looking mild for the weekend. No surprises expected.
Predicted Sentiment: negative
Actual Sentiment: neutral
```

[6]:
```
# use scikit-learn's classification report
print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.79      0.94      0.86        16
     neutral       0.78      0.44      0.56        16
    positive       0.68      0.83      0.75        18

    accuracy                           0.74        50
   macro avg       0.75      0.74      0.72        50
weighted avg       0.75      0.74      0.72        50
```

1.2.5 [4 points] Question 4:

Run VADER on the set of airline tweets with the following settings:

- Run VADER (as it is) on the set of airline tweets
- Run VADER on the set of airline tweets after having lemmatized the text
- Run VADER on the set of airline tweets with only adjectives
- Run VADER on the set of airline tweets with only adjectives and after having lemmatized the text
- Run VADER on the set of airline tweets with only nouns
- Run VADER on the set of airline tweets with only nouns and after having lemmatized the text
- Run VADER on the set of airline tweets with only verbs
- Run VADER on the set of airline tweets with only verbs and after having lemmatized the text

[1 point] a. Generate for all separate experiments the classification report, i.e., Precision, Recall, and F1 scores per category as well as micro and macro averages. **Use a different code cell (or multiple code cells) for each experiment.**

[3 points] b. Compare the scores and explain what they tell you.

Does lemmatisation help? Explain why or why not.

- Comparing the original airline tweet text with the lemmatized version, we observe only minor differences, with accuracy decreasing slightly from 0.63 to 0.62. For tweets containing only adjectives, the accuracy remains 0.50 for both lemmatized and non-lemmatized versions. Similarly, for tweets with only nouns, the accuracy stays consistent at 0.42 in both cases. The same thing holds for verbs, where accuracy remains 0.47, regardless of lemmatization.

This suggests that lemmatization does not significantly impact sentiment classification in this case. This can be explained by the fact that VADER primarily relies on predefined lexicons so lemmatization doesn't improve classification significantly.

Are all parts of speech equally important for sentiment analysis? Explain why or why not.

- Adjectives achieve the highest accuracy (0.50), suggesting they play a significant role in conveying sentiment, likely because they describe more emotions and opinions. Verbs performance is slightly less acurrate, with an accuracy of 0.47, indicating they contribute to sentiment but in some cases some verbs are neutral. And lastly, Nouns show the weakest performance, with an accuracy of 0.42, this can be explained by the fact that nouns are mostly neutral and are less effective in capturing sentiments.

[7]:
```
import nltk
import spacy
nlp = spacy.load('en_core_web_sm')

def run_vader(textual_unit,
              lemmatize=False,
              parts_of_speech_to_consider=None,
              verbose=0):
    """
    Run VADER on a sentence from spacy

    :param str textual unit: a textual unit, e.g., sentence, sentences (one␣
 ↪string)
    (by looping over doc.sents)
    :param bool lemmatize: If True, provide lemmas to VADER instead of words
    :param set parts_of_speech_to_consider:
    -None or empty set: all parts of speech are provided
    -non-empty set: only these parts of speech are considered.
    :param int verbose: if set to 1, information is printed
    about input and output

    :rtype: dict
    :return: vader output dict
    """
    doc = nlp(textual_unit)

    input_to_vader = []

    for sent in doc.sents:
        for token in sent:

            to_add = token.text

            if lemmatize:
                to_add = token.lemma_
```

```
                if to_add == '-PRON-':
                    to_add = token.text

            if parts_of_speech_to_consider:
                if token.pos_ in parts_of_speech_to_consider:
                    input_to_vader.append(to_add)
            else:
                input_to_vader.append(to_add)

    scores = vader_model.polarity_scores(' '.join(input_to_vader))

    if verbose >= 1:
        print('INPUT SENTENCE', sent)
        print('INPUT TO VADER', input_to_vader)
        print('VADER OUTPUT', scores)

    return scores
```

[8]:
```
#Run VADER (as it is) on the set of airline tweets
import os
import pathlib
cwd = pathlib.Path.cwd()
airline_tweets_folder = cwd.joinpath('airlinetweets')
rootdir = str(airline_tweets_folder)

def find_gold_label(subdir):
    if 'negative' in subdir.lower():
        gold_label = 'negative'
    elif 'positive' in subdir.lower():
        gold_label = 'positive'
    elif 'neutral' in subdir.lower():
        gold_label = 'neutral'
    return gold_label

all_vader_output = []
gold = []
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                scores = run_vader(content,verbose=1)
                vader_label = vader_output_to_label(scores)
                all_vader_output.append(vader_label)
```

```
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER ['@VirginAmerica', 'my', 'group', 'got', 'their', 'Cancelled',
'Flightlation', 'fees', 'waived', 'but', 'I', 'ca', "n't", 'because', 'my',
'ticket', 'is', 'booked', 'for', '2/18', '?', 'Your', 'reps', 'were', 'no',
'help', 'either', ' ']
VADER OUTPUT {'neg': 0.144, 'neu': 0.737, 'pos': 0.119, 'compound': 0.0644}
Predicted Sentiment: positive
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER ['"', '@VirginAmerica', 'As', 'one', 'of', 'the', 'travelers',
'affected', 'by', 'the', 'Boston', 'storm']
VADER OUTPUT {'neg': 0.138, 'neu': 0.862, 'pos': 0.0, 'compound': -0.1531}
Predicted Sentiment: negative
Actual Sentiment: negative

Tweet: @VirginAmerica Grouping Virgin in with the others now. BOS weather has
exposed their actual Cus Serv model. Never Flight Booking Problems with Virgin
again!
INPUT SENTENCE Never Flight Booking Problems with Virgin again!
INPUT TO VADER ['@VirginAmerica', 'Grouping', 'Virgin', 'in', 'with', 'the',
'others', 'now', '.', 'BOS', 'weather', 'has', 'exposed', 'their', 'actual',
'Cus', 'Serv', 'model', '.', 'Never', 'Flight', 'Booking', 'Problems', 'with',
'Virgin', 'again', '!']
VADER OUTPUT {'neg': 0.05, 'neu': 0.851, 'pos': 0.099, 'compound': 0.3071}
Predicted Sentiment: positive
Actual Sentiment: negative

Tweet: "@VirginAmerica While other airlines weren't Cancelled Flighting flights
into BOS
INPUT SENTENCE "@VirginAmerica While other airlines weren't Cancelled Flighting
flights into BOS
INPUT TO VADER ['"', '@VirginAmerica', 'While', 'other', 'airlines', 'were',
"n't", 'Cancelled', 'Flighting', 'flights', 'into', 'BOS']
VADER OUTPUT {'neg': 0.0, 'neu': 0.852, 'pos': 0.148, 'compound': 0.1877}
Predicted Sentiment: positive
Actual Sentiment: negative
```

```
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @SouthwestAir thank you so much completely made things right!
INPUT SENTENCE @SouthwestAir thank you so much completely made things right!
INPUT TO VADER ['@SouthwestAir', 'thank', 'you', 'so', 'much', 'completely',
'made', 'things', 'right', '!']
VADER OUTPUT {'neg': 0.0, 'neu': 0.741, 'pos': 0.259, 'compound': 0.4199}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @AmericanAir thank you!
INPUT SENTENCE @AmericanAir thank you!
INPUT TO VADER ['@AmericanAir', 'thank', 'you', '!']
VADER OUTPUT {'neg': 0.0, 'neu': 0.417, 'pos': 0.583, 'compound': 0.4199}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @USAirways we were moved to a delta direct. Thank you for the
accommodations!
INPUT SENTENCE Thank you for the accommodations!
INPUT TO VADER ['@USAirways', 'we', 'were', 'moved', 'to', 'a', 'delta',
'direct', '.', 'Thank', 'you', 'for', 'the', 'accommodations', '!']
VADER OUTPUT {'neg': 0.0, 'neu': 0.798, 'pos': 0.202, 'compound': 0.4199}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['@united', 'Thanks', 'for', 'taking', 'care', 'of', 'that',
'MR', '!', '!', 'Happy', 'customer', '.']
VADER OUTPUT {'neg': 0.0, 'neu': 0.403, 'pos': 0.597, 'compound': 0.8856}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER ['@united', 'thanks']
VADER OUTPUT {'neg': 0.0, 'neu': 0.256, 'pos': 0.744, 'compound': 0.4404}
Predicted Sentiment: positive
Actual Sentiment: positive
```

[9]:
```
# VADER (as it is) on the set of airline tweets
print(classification_report(gold,all_vader_output))
```

```
              precision    recall  f1-score   support
```

```
    negative       0.80      0.51      0.63      1750
     neutral       0.60      0.51      0.55      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.63      4755
   macro avg       0.65      0.64      0.62      4755
weighted avg       0.66      0.63      0.62      4755
```

[10]:
```
#Run VADER on the set of airline tweets after having lemmatized the text
all_vader_output = []
gold = []
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                lemma = run_vader(content, lemmatize=True, verbose=1)
                vader_label = vader_output_to_label(lemma)
                all_vader_output.append(vader_label)
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER ['@VirginAmerica', 'my', 'group', 'get', 'their', 'Cancelled',
'Flightlation', 'fee', 'waive', 'but', 'I', 'can', 'not', 'because', 'my',
'ticket', 'be', 'book', 'for', '2/18', '?', 'your', 'rep', 'be', 'no', 'help',
'either', ' ']
VADER OUTPUT {'neg': 0.144, 'neu': 0.737, 'pos': 0.119, 'compound': 0.0644}
Predicted Sentiment: positive
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER ['"', '@VirginAmerica', 'as', 'one', 'of', 'the', 'traveler',
'affect', 'by', 'the', 'Boston', 'storm']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative
```

```
INPUT SENTENCE @SouthwestAir thank you so much completely made things right!
INPUT TO VADER ['@southwestair', 'thank', 'you', 'so', 'much', 'completely',
'make', 'thing', 'right', '!']
VADER OUTPUT {'neg': 0.0, 'neu': 0.741, 'pos': 0.259, 'compound': 0.4199}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @AmericanAir thank you!
INPUT SENTENCE @AmericanAir thank you!
INPUT TO VADER ['@americanair', 'thank', 'you', '!']
VADER OUTPUT {'neg': 0.0, 'neu': 0.417, 'pos': 0.583, 'compound': 0.4199}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @USAirways we were moved to a delta direct. Thank you for the
accommodations!
INPUT SENTENCE Thank you for the accommodations!
INPUT TO VADER ['@usairway', 'we', 'be', 'move', 'to', 'a', 'delta', 'direct',
'.', 'thank', 'you', 'for', 'the', 'accommodation', '!']
VADER OUTPUT {'neg': 0.0, 'neu': 0.798, 'pos': 0.202, 'compound': 0.4199}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['@united', 'thank', 'for', 'take', 'care', 'of', 'that', 'MR',
'!', '!', 'happy', 'customer', '.']
VADER OUTPUT {'neg': 0.0, 'neu': 0.412, 'pos': 0.588, 'compound': 0.8745}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER ['@unite', 'thank']
VADER OUTPUT {'neg': 0.0, 'neu': 0.286, 'pos': 0.714, 'compound': 0.3612}
Predicted Sentiment: positive
Actual Sentiment: positive
```

[11]:
```
# VADER on the set of airline tweets after having lemmatized the text
print(classification_report(gold,all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.79      0.52      0.63      1750
     neutral       0.60      0.49      0.54      1515
    positive       0.56      0.88      0.68      1490
```

```
    accuracy                           0.62      4755
   macro avg       0.65      0.63      0.62      4755
weighted avg       0.65      0.62      0.62      4755
```

[12]:
```
#Run VADER on the set of airline tweets with only adjectives
all_vader_output = []
gold = []
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                adjectives = run_vader(content,
                            parts_of_speech_to_consider={'ADJ'}, verbose=1)
                vader_label = vader_output_to_label(adjectives)
                all_vader_output.append(vader_label)
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: @VirginAmerica Grouping Virgin in with the others now. BOS weather has
exposed their actual Cus Serv model. Never Flight Booking Problems with Virgin
again!
INPUT SENTENCE Never Flight Booking Problems with Virgin again!
INPUT TO VADER ['actual']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
```

```
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @AmericanAir thank you!
INPUT SENTENCE @AmericanAir thank you!
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @USAirways we were moved to a delta direct. Thank you for the
accommodations!
INPUT SENTENCE Thank you for the accommodations!
INPUT TO VADER ['direct']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['@united', 'Happy']
VADER OUTPUT {'neg': 0.0, 'neu': 0.213, 'pos': 0.787, 'compound': 0.5719}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive
```

[13]:
```
# VADER on the set of airline tweets with only adjectives
print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.86      0.20      0.33      1750
     neutral       0.40      0.89      0.55      1515
    positive       0.67      0.44      0.53      1490

    accuracy                           0.50      4755
   macro avg       0.64      0.51      0.47      4755
weighted avg       0.65      0.50      0.46      4755
```

[14]:
```
#Run VADER on the set of airline tweets with only adjectives and after having␣
 ↪lemmatized the text
all_vader_output = []
gold = []
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                adjectives = run_vader(content,lemmatize=True,␣
 ↪parts_of_speech_to_consider={'ADJ'},verbose=1)
                vader_label = vader_output_to_label(adjectives)
                all_vader_output.append(vader_label)
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: @VirginAmerica Grouping Virgin in with the others now. BOS weather has
exposed their actual Cus Serv model. Never Flight Booking Problems with Virgin
again!
INPUT SENTENCE Never Flight Booking Problems with Virgin again!
INPUT TO VADER ['actual']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: "@VirginAmerica While other airlines weren't Cancelled Flighting flights
into BOS
```

```
Tweet: @AmericanAir thank you!
INPUT SENTENCE @AmericanAir thank you!
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @USAirways we were moved to a delta direct. Thank you for the
accommodations!
INPUT SENTENCE Thank you for the accommodations!
INPUT TO VADER ['direct']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['@united', 'happy']
VADER OUTPUT {'neg': 0.0, 'neu': 0.213, 'pos': 0.787, 'compound': 0.5719}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive
```

[15]:
```
# VADER on the set of airline tweets with only adjectives and after having␣
↪lemmatized the text
print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.86      0.20      0.33      1750
     neutral       0.40      0.89      0.55      1515
    positive       0.67      0.44      0.53      1490

    accuracy                           0.50      4755
   macro avg       0.64      0.51      0.47      4755
weighted avg       0.65      0.50      0.46      4755
```

[16]:
```
#Run VADER on the set of airline tweets with only nouns
all_vader_output = []
gold = []
```

```
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                nouns = run_vader(content,
                            parts_of_speech_to_consider={'NOUN'},
                            verbose=1)
                vader_label = vader_output_to_label(nouns)
                all_vader_output.append(vader_label)
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER ['group', 'fees', 'ticket', 'reps', 'help']
VADER OUTPUT {'neg': 0.0, 'neu': 0.597, 'pos': 0.403, 'compound': 0.4019}
Predicted Sentiment: positive
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER ['travelers', 'storm']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: @VirginAmerica Grouping Virgin in with the others now. BOS weather has
exposed their actual Cus Serv model. Never Flight Booking Problems with Virgin
again!
INPUT SENTENCE Never Flight Booking Problems with Virgin again!
INPUT TO VADER ['others', 'weather', 'model']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: "@VirginAmerica While other airlines weren't Cancelled Flighting flights
into BOS
INPUT SENTENCE "@VirginAmerica While other airlines weren't Cancelled Flighting
flights into BOS
INPUT TO VADER ['@VirginAmerica', 'airlines', 'flights']
```

```
accommodations!
INPUT SENTENCE Thank you for the accommodations!
INPUT TO VADER ['@USAirways', 'delta', 'accommodations']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['Thanks', 'care', 'customer']
VADER OUTPUT {'neg': 0.0, 'neu': 0.141, 'pos': 0.859, 'compound': 0.7269}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER ['thanks']
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.4404}
Predicted Sentiment: positive
Actual Sentiment: positive
```

[17]:
```
#Run VADER on the set of airline tweets with only nouns
print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.73      0.14      0.23      1750
     neutral       0.36      0.82      0.50      1515
    positive       0.53      0.35      0.42      1490

    accuracy                           0.42      4755
   macro avg       0.54      0.44      0.39      4755
weighted avg       0.55      0.42      0.38      4755
```

[18]:
```
#Run VADER on the set of airline tweets with only nouns and after having␣
 ↪lemmatized the text
all_vader_output = []
gold = []
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                nouns = run_vader(content,
```

```
                                lemmatize=True,
                                parts_of_speech_to_consider={'NOUN'},
                                verbose=1)
                vader_label = vader_output_to_label(nouns)
                all_vader_output.append(vader_label)
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER ['group', 'fee', 'ticket', 'rep', 'help']
VADER OUTPUT {'neg': 0.0, 'neu': 0.597, 'pos': 0.403, 'compound': 0.4019}
Predicted Sentiment: positive
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER ['traveler', 'storm']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: @VirginAmerica Grouping Virgin in with the others now. BOS weather has
exposed their actual Cus Serv model. Never Flight Booking Problems with Virgin
again!
INPUT SENTENCE Never Flight Booking Problems with Virgin again!
INPUT TO VADER ['other', 'weather', 'model']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: "@VirginAmerica While other airlines weren't Cancelled Flighting flights
into BOS
INPUT SENTENCE "@VirginAmerica While other airlines weren't Cancelled Flighting
flights into BOS
INPUT TO VADER ['@VirginAmerica', 'airline', 'flight']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: @VirginAmerica started my flight with a scolding for using an overhead
bin that was then offered to the person seated next to me.
INPUT SENTENCE @VirginAmerica started my flight with a scolding for using an
```

```
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @SouthwestAir thank you so much completely made things right!
INPUT SENTENCE @SouthwestAir thank you so much completely made things right!
INPUT TO VADER ['thing']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @AmericanAir thank you!
INPUT SENTENCE @AmericanAir thank you!
INPUT TO VADER []
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @USAirways we were moved to a delta direct. Thank you for the
accommodations!
INPUT SENTENCE Thank you for the accommodations!
INPUT TO VADER ['@usairway', 'delta', 'accommodation']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['thank', 'care', 'customer']
VADER OUTPUT {'neg': 0.0, 'neu': 0.149, 'pos': 0.851, 'compound': 0.6908}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER ['thank']
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.3612}
Predicted Sentiment: positive
Actual Sentiment: positive
```

[19]:
```
#Run VADER on the set of airline tweets with only nouns and after having␣
 ↪lemmatized the text
print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.71      0.15      0.25      1750
     neutral       0.36      0.81      0.50      1515
```

```
    positive       0.52      0.34      0.41      1490

    accuracy                           0.42      4755
   macro avg       0.53      0.44      0.39      4755
weighted avg       0.54      0.42      0.38      4755
```

[20]:
```
#Run VADER on the set of airline tweets with only verbs
all_vader_output = []
gold = []
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                verbs = run_vader(content,
                            parts_of_speech_to_consider={'VERB'},
                            verbose=1)
                vader_label = vader_output_to_label(verbs)
                all_vader_output.append(vader_label)
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER ['got', 'waived', 'booked']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER ['affected']
VADER OUTPUT {'neg': 1.0, 'neu': 0.0, 'pos': 0.0, 'compound': -0.1531}
Predicted Sentiment: negative
Actual Sentiment: negative

Tweet: @VirginAmerica Grouping Virgin in with the others now. BOS weather has
exposed their actual Cus Serv model. Never Flight Booking Problems with Virgin
again!
INPUT SENTENCE Never Flight Booking Problems with Virgin again!
```

```
INPUT TO VADER ['moved', 'Thank']
VADER OUTPUT {'neg': 0.0, 'neu': 0.286, 'pos': 0.714, 'compound': 0.3612}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['taking']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER ['@united']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive
```

[21]:
```
#Run VADER on the set of airline tweets with only verbs
print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.79      0.29      0.42      1750
     neutral       0.38      0.81      0.52      1515
    positive       0.57      0.35      0.43      1490

    accuracy                           0.47      4755
   macro avg       0.58      0.48      0.46      4755
weighted avg       0.59      0.47      0.46      4755
```

[ ]:
```
#Run VADER on the set of airline tweets with only verbs and after having␣
 ↪lemmatized the text
all_vader_output = []
gold = []
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        x = os.path.join(subdir, file)
        if x.endswith('.txt'):
            with open(x, 'r', encoding='utf-8') as f:
                content = f.read()
                print(f"Tweet: {content}")
                verbs = run_vader(content,
                            lemmatize=True,
                            parts_of_speech_to_consider={'VERB'},
```

```
                        verbose=1)
                vader_label = vader_output_to_label(verbs)
                all_vader_output.append(vader_label)
                gold_label = find_gold_label(subdir)
                gold.append(gold_label)
                print(f"Predicted Sentiment: {vader_label}")
                print(f"Actual Sentiment: {gold_label}")
                print()
```

```
Tweet: @VirginAmerica my group got their Cancelled Flightlation fees waived but
I can't because my ticket is booked for 2/18? Your reps were no help either
INPUT SENTENCE Your reps were no help either
INPUT TO VADER ['get', 'waive', 'book']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: "@VirginAmerica As one of the travelers affected by the Boston storm
INPUT SENTENCE "@VirginAmerica As one of the travelers affected by the Boston
storm
INPUT TO VADER ['affect']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: negative

Tweet: @VirginAmerica Grouping Virgin in with the others now. BOS weather has
exposed their actual Cus Serv model. Never Flight Booking Problems with Virgin
again!
INPUT SENTENCE Never Flight Booking Problems with Virgin again!
INPUT TO VADER ['expose']
VADER OUTPUT {'neg': 1.0, 'neu': 0.0, 'pos': 0.0, 'compound': -0.1531}
Predicted Sentiment: negative
Actual Sentiment: negative

Tweet: "@VirginAmerica While other airlines weren't Cancelled Flighting flights
into BOS
INPUT SENTENCE "@VirginAmerica While other airlines weren't Cancelled Flighting
flights into BOS
INPUT TO VADER ['cancel', 'flight']
VADER OUTPUT {'neg': 0.667, 'neu': 0.333, 'pos': 0.0, 'compound': -0.25}
Predicted Sentiment: negative
Actual Sentiment: negative

Tweet: @VirginAmerica started my flight with a scolding for using an overhead
bin that was then offered to the person seated next to me.
INPUT SENTENCE @VirginAmerica started my flight with a scolding for using an
overhead bin that was then offered to the person seated next to me.
INPUT TO VADER ['start', 'use', 'offer', 'seat']
```

```
Tweet: @AmericanAir thank you!
INPUT SENTENCE @AmericanAir thank you!
INPUT TO VADER ['thank']
VADER OUTPUT {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.3612}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @USAirways we were moved to a delta direct. Thank you for the
accommodations!
INPUT SENTENCE Thank you for the accommodations!
INPUT TO VADER ['move', 'thank']
VADER OUTPUT {'neg': 0.0, 'neu': 0.286, 'pos': 0.714, 'compound': 0.3612}
Predicted Sentiment: positive
Actual Sentiment: positive

Tweet: @united Thanks for taking care of that MR!! Happy customer.
INPUT SENTENCE Happy customer.
INPUT TO VADER ['take']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive

Tweet: @united thanks
INPUT SENTENCE @united thanks
INPUT TO VADER ['@unite']
VADER OUTPUT {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Predicted Sentiment: neutral
Actual Sentiment: positive
```

[ ]:
```
#Run VADER on the set of airline tweets with only verbs and after having␣
 ↪lemmatized the text
print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.75      0.29      0.42      1750
     neutral       0.38      0.78      0.51      1515
    positive       0.57      0.36      0.44      1490

    accuracy                           0.47      4755
   macro avg       0.57      0.48      0.46      4755
weighted avg       0.58      0.47      0.46      4755
```

1.3 Part II: scikit-learn assignments

1.3.1 [4 points] Question 5

Train the scikit-learn classifier (Naive Bayes) using the airline tweets.

- Train the model on the airline tweets with 80% training and 20% test set and default settings (TF-IDF representation, min_df=2)
- Train with different settings:
  - with respect to vectorizing: TF-IDF ('airline_tfidf') vs. Bag of words representation ('airline_count')
  - with respect to the frequency threshold (min_df). Carry out experiments with increasing values for document frequency (min_df = 2; min_df = 5; min_df =10)
- [1 point] a. Generate a classification_report for all experiments
- [3 points] b. Look at the results of the experiments with the different settings and try to explain why they differ:
  - which category performs best, is this the case for any setting?
    The model with bag of words representation performs the best, specifically with a document frequency value of 2. The bag of word representation counts the frequency of words which occur in the sentences, but does not consider the importance of words. On the other hand, with TF-IDF the importance of a word is based on the frequency of the word in the sentences and words that occur accross most of the documents have lower importance (i.e., common words). One reason why TF-IDF performs worse is that some frequent words like "delay" and "wait" may have been given less importance, despite still being important for determining the sentiment of sentences.
  - does the frequency threshold affect the scores? Why or why not according to you?
    Yes, the frequency threshold does affect the scores; the performance is better when the frequency threshold is smaller. A lower frequency threshold allows important, but less frequent words to be included. While these words do not appear as often, but they can still be important indicators for the sentiment.

[24]:
```
from sklearn.datasets import load_files
import nltk
import pathlib
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from nltk.corpus import stopwords
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

cwd = pathlib.Path.cwd()
airline_tweets_folder = cwd.joinpath('airlinetweets')
path = str(airline_tweets_folder)
airline_tweets_train = load_files(path)

# TF-IDF, min_df=2
#Feature extraction
airline_vec = CountVectorizer(min_df=2,
```

```
                              tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                              stop_words=stopwords.words('english')) # stopwords␣
 ↪are removed
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)
tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

#Training with tf-idf model
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_tfidf, # the tf-idf model
    airline_tweets_train.target, # the category values for each tweet
    test_size = 0.20 # we use 80% for training and 20% for testing
    )
clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

print(classification_report(y_test, y_pred))
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:517: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:402: UserWarning: Your stop_words
may be inconsistent with your preprocessing. Tokenizing the stop words generated
tokens ["'d", "'ll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't",
'need', 'sha', 'wo', 'would'] not in stop_words.
  warnings.warn(
```

	precision	recall	f1-score	support
0	0.82	0.91	0.86	361
1	0.88	0.70	0.78	317
2	0.79	0.86	0.82	273
accuracy			0.83	951
macro avg	0.83	0.83	0.82	951
weighted avg	0.83	0.83	0.83	951

[25]:
```
# TF-IDF, min_df=5
#Feature extraction
airline_vec = CountVectorizer(min_df=5,
                              tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                              stop_words=stopwords.words('english')) # stopwords␣
 ↪are removed
```

```
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)
tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

#Training with tf-idf model
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_tfidf, # the tf-idf model
    airline_tweets_train.target, # the category values for each tweet
    test_size = 0.20 # we use 80% for training and 20% for testing
    )
clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

print(classification_report(y_test, y_pred))
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:517: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:402: UserWarning: Your stop_words
may be inconsistent with your preprocessing. Tokenizing the stop words generated
tokens ["'d", "'ll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't",
'need', 'sha', 'wo', 'would'] not in stop_words.
  warnings.warn(
```

	precision	recall	f1-score	support
0	0.84	0.89	0.87	354
1	0.81	0.75	0.78	290
2	0.85	0.84	0.84	307
accuracy			0.83	951
macro avg	0.83	0.83	0.83	951
weighted avg	0.83	0.83	0.83	951

[26]:
```
# TF-IDF, min_df=10
#Feature extraction
airline_vec = CountVectorizer(min_df=10,
                              tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                              stop_words=stopwords.words('english')) # stopwords␣
 ↪are removed
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)
tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)
```

```
#Training with tf-idf model
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_tfidf, # the tf-idf model
    airline_tweets_train.target, # the category values for each tweet
    test_size = 0.20 # we use 80% for training and 20% for testing
    )
clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

print(classification_report(y_test, y_pred))
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:517: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:402: UserWarning: Your stop_words
may be inconsistent with your preprocessing. Tokenizing the stop words generated
tokens ["'d", "'ll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't",
'need', 'sha', 'wo', 'would'] not in stop_words.
  warnings.warn(
```

	precision	recall	f1-score	support
0	0.82	0.88	0.85	350
1	0.78	0.73	0.76	287
2	0.84	0.82	0.83	314
accuracy			0.82	951
macro avg	0.81	0.81	0.81	951
weighted avg	0.82	0.82	0.82	951

[27]:
```
# Bag of words representation, min_df=2
airline_vec = CountVectorizer(min_df=2,
                              tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                              stop_words=stopwords.words('english')) # stopwords␣
 ↪are removed
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

#Training with airline_counts model
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # the airline_counts model
    airline_tweets_train.target, # the category values for each tweet
    test_size = 0.20 # we use 80% for training and 20% for testing
    )
clf = MultinomialNB().fit(docs_train, y_train)
```

```
y_pred = clf.predict(docs_test)

print(classification_report(y_test, y_pred))
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:517: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:402: UserWarning: Your stop_words
may be inconsistent with your preprocessing. Tokenizing the stop words generated
tokens ["'d", "'ll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't",
'need', 'sha', 'wo', 'would'] not in stop_words.
  warnings.warn(
```

	precision	recall	f1-score	support
0	0.89	0.90	0.90	367
1	0.87	0.76	0.81	295
2	0.83	0.92	0.87	289
accuracy			0.86	951
macro avg	0.86	0.86	0.86	951
weighted avg	0.87	0.86	0.86	951

[28]:
```
# Bag of words representation, min_df=5
airline_vec = CountVectorizer(min_df=5,
                              tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                              stop_words=stopwords.words('english')) # stopwords␣
 ↪are removed
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

#Training with airline_counts model
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # the airline_counts model
    airline_tweets_train.target, # the category values for each tweet
    test_size = 0.20 # we use 80% for training and 20% for testing
    )
clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

print(classification_report(y_test, y_pred))
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:517: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:402: UserWarning: Your stop_words
may be inconsistent with your preprocessing. Tokenizing the stop words generated
tokens ["'d", "'ll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't",
'need', 'sha', 'wo', 'would'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

           0       0.83      0.88      0.85       350
           1       0.80      0.74      0.77       295
           2       0.83      0.84      0.83       306

    accuracy                           0.82       951
   macro avg       0.82      0.82      0.82       951
weighted avg       0.82      0.82      0.82       951
```

[29]:
```
# Bag of words representation, min_df=10
airline_vec = CountVectorizer(min_df=10,
                              tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                              stop_words=stopwords.words('english')) # stopwords␣
 ↪are removed
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

#Training with airline_counts model
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # the airline_counts model
    airline_tweets_train.target, # the category values for each tweet
    test_size = 0.20 # we use 80% for training and 20% for testing
    )
clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

print(classification_report(y_test, y_pred))
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:517: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:402: UserWarning: Your stop_words
may be inconsistent with your preprocessing. Tokenizing the stop words generated
tokens ["'d", "'ll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't",
'need', 'sha', 'wo', 'would'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support
```

```
              0       0.82      0.88      0.85       314
              1       0.81      0.75      0.78       329
              2       0.80      0.81      0.81       308

    accuracy                            0.81       951
   macro avg       0.81      0.81      0.81       951
weighted avg       0.81      0.81      0.81       951
```

1.3.2 [4 points] Question 6: Inspecting the best scoring features

- Train the scikit-learn classifier (Naive Bayes) model with the following settings (airline tweets 80% training and 20% test; Bag of words representation ('airline_count'), min_df=2)
- [1 point] a. Generate the list of best scoring features per class (see function **important_features_per_class** below) [1 point]
- [3 points] b. Look at the lists and consider the following issues:
  - [1 point] Which features did you expect for each separate class and why?
    - For negative list, I expected words related to complaints, dissatisfaction. For neutral list, I expected general travel related terms. And for positive list, I expected words conveying appreciation and satisfaction.
  - [1 point] Which features did you not expect and why ?
    - In the negative list the occurence of airline names was unexpected, this could be axplained by the fact that when people write reviews they often mention the airline name. Furthermore, the word 'Thanks' is also present in the negative list which is also surprising as we expect it to be founs under the positive list. In the neutral list we find the word 'Thanks' again and the words 'cancelled' which we would expect o belong to the negative list.
  - [1 point] The list contains all kinds of words such as names of airlines, punctuation, numbers and content words (e.g., 'delay' and 'bad'). Which words would you remove or keep when trying to improve the model and why?
    - To improve the model we would remove mainly symboles such as (@, #, &, http) as well as number (2,3,4) as they do not convey any sentiments and thus do not contribute to the classification. we would keep however words that convey strong sentiments like 'delay', 'worst', 'great', 'amazing', and 'thank' as well as words like 'flight', 'customer', and 'service' that are neutral but relevant since they provide context.

[30]:
```
airline_vec = CountVectorizer(min_df=2,
                              tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                              stop_words=stopwords.words('english')) # stopwords␣
 ↪are removed
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

#Training with airline_counts model
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # the airline_counts model
```

```
    airline_tweets_train.target, # the category values for each tweet
    test_size = 0.20 # we use 80% for training and 20% for testing
    )
clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)
```

```
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:517: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\amina\anaconda3\envs\text-mining\Lib\site-
packages\sklearn\feature_extraction\text.py:402: UserWarning: Your stop_words
may be inconsistent with your preprocessing. Tokenizing the stop words generated
tokens ["'d", "'ll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't",
'need', 'sha', 'wo', 'would'] not in stop_words.
  warnings.warn(
```

[31]:
```
def important_features_per_class(vectorizer,classifier,n=80):
    class_labels = classifier.classes_
    feature_names =vectorizer.get_feature_names_out()
    topn_class1 = sorted(zip(classifier.feature_count_[0],␣
 ↪feature_names),reverse=True)[:n]
    topn_class2 = sorted(zip(classifier.feature_count_[1],␣
 ↪feature_names),reverse=True)[:n]
    topn_class3 = sorted(zip(classifier.feature_count_[2],␣
 ↪feature_names),reverse=True)[:n]
    print("Important words in negative documents")
    for coef, feat in topn_class1:
        print(class_labels[0], coef, feat)
    print("-----------------------------------------")
    print("Important words in neutral documents")
    for coef, feat in topn_class2:
        print(class_labels[1], coef, feat)
    print("-----------------------------------------")
    print("Important words in positive documents")
    for coef, feat in topn_class3:
        print(class_labels[2], coef, feat)

# example of how to call from notebook:
important_features_per_class(airline_vec, clf)
```

```
Important words in negative documents
0 1508.0 @
0 1379.0 united
0 1241.0 .
0 428.0 ``
0 389.0 flight
0 377.0 ?
```

0 369.0 !
0 340.0 #
0 220.0 n't
0 152.0 ''
0 121.0 's
0 114.0 :
0 113.0 virginamerica
0 108.0 service
0 103.0 get
0 95.0 cancelled
0 91.0 plane
0 85.0 customer
0 84.0 time
0 83.0 delayed
0 83.0 bag
0 74.0 -
0 73.0 hours
0 72.0 'm
0 71.0 http
0 69.0 ;
0 66.0 …
0 64.0 still
0 64.0 &
0 63.0 gate
0 60.0 airline
0 59.0 would
0 59.0 late
0 59.0 flights
0 57.0 hour
0 56.0 one
0 56.0 help
0 56.0 ca
0 54.0 amp
0 54.0 2
0 52.0 delay
0 50.0 worst
0 50.0 $
0 49.0 never
0 48.0 like
0 46.0 waiting
0 45.0 've
0 44.0 (
0 42.0 flightled
0 41.0 us
0 41.0 lost
0 40.0 3
0 40.0)
0 39.0 wait

0 39.0 really
0 39.0 luggage
0 39.0 fly
0 39.0 ever
0 38.0 due
0 37.0 thanks
0 37.0 people
0 37.0 day
0 37.0 back
0 35.0 bags
0 34.0 trying
0 34.0 seat
0 34.0 another
0 33.0 hold
0 32.0 baggage
0 32.0 4
0 31.0 last
0 31.0 days
0 30.0 problems
0 30.0 got
0 30.0 going
0 30.0 check
0 29.0 ticket
0 29.0 terrible
0 29.0 crew
0 29.0 airport

Important words in neutral documents
1 1385.0 @
1 499.0 .
1 495.0 ?
1 293.0 jetblue
1 277.0 :
1 260.0 southwestair
1 256.0 united
1 247.0 ``
1 235.0 flight
1 226.0 #
1 188.0 americanair
1 179.0 http
1 162.0 !
1 152.0 usairways
1 134.0 's
1 80.0 get
1 75.0 virginamerica
1 73.0 -
1 70.0 ''
1 69.0 flights

1 64.0 please
1 61.0 help
1 56.0)
1 52.0 need
1 47.0 (
1 46.0 n't
1 45.0 ;
1 43.0 …
1 42.0 dm
1 40.0 would
1 40.0 us
1 39.0 &
1 38.0 fleet
1 38.0 fleek
1 36.0 "
1 36.0 know
1 35.0 way
1 34.0 tomorrow
1 34.0 thanks
1 32.0 flying
1 32.0 amp
1 31.0 "
1 31.0 hi
1 30.0 could
1 30.0 change
1 30.0 cancelled
1 30.0 'm
1 29.0 number
1 28.0 like
1 28.0 fly
1 27.0 new
1 26.0 one
1 24.0 today
1 23.0 travel
1 23.0 time
1 23.0 see
1 23.0 go
1 23.0 check
1 23.0 airport
1 22.0 ticket
1 21.0 reservation
1 21.0 next
1 21.0 destinationdragons
1 21.0 add
1 20.0 sent
1 20.0 rt
1 20.0 question
1 20.0 follow

```
1 19.0 want
1 19.0 tickets
1 19.0 seat
1 19.0 booked
1 19.0 back
1 18.0 weather
1 18.0 trip
1 18.0 make
1 18.0 going
1 18.0 first
1 18.0 chance
1 18.0 bag
----------------------------------------
Important words in positive documents
2 1352.0 @
2 1081.0 !
2 785.0 .
2 321.0 #
2 313.0 southwestair
2 287.0 jetblue
2 286.0 thanks
2 258.0 united
2 245.0 thank
2 231.0 ``
2 178.0 flight
2 169.0 americanair
2 164.0 :
2 138.0 great
2 137.0 usairways
2 100.0 service
2 87.0 virginamerica
2 86.0 )
2 75.0 http
2 74.0 love
2 71.0 customer
2 69.0 guys
2 66.0 best
2 62.0 much
2 62.0 ;
2 60.0 's
2 57.0 awesome
2 53.0 good
2 53.0 -
2 50.0 airline
2 45.0 time
2 45.0 &
2 44.0 amazing
2 42.0 got
```

2 42.0 get
2 41.0 us
2 41.0 today
2 41.0 n't
2 40.0 crew
2 37.0 help
2 36.0 gate
2 36.0 amp
2 35.0 made
2 33.0 …
2 32.0 fly
2 31.0 flying
2 31.0 appreciate
2 28.0 see
2 28.0 back
2 27.0 response
2 27.0 'm
2 26.0 u
2 25.0 work
2 25.0 day
2 25.0 ?
2 25.0 're
2 24.0 well
2 24.0 team
2 24.0 nice
2 24.0 new
2 24.0 flights
2 23.0 yes
2 23.0 tonight
2 23.0 southwest
2 23.0 like
2 23.0 job
2 23.0 home
2 23.0 ever
2 23.0 always
2 23.0 ''
2 22.0 staff
2 22.0 plane
2 22.0 helpful
2 21.0 would
2 21.0 happy
2 21.0 first
2 20.0 really
2 20.0 know
2 19.0 way
2 19.0 please

1.3.3 [Optional! (will not be graded)] Question 7

Train the model on airline tweets and test it on your own set of tweets + Train the model with the following settings (airline tweets 80% training and 20% test; Bag of words representation ('airline_count'), min_df=2) + Apply the model on your own set of tweets and generate the classification report * [1 point] a. Carry out a quantitative analysis. * [1 point] b. Carry out an error analysis on 10 correctly and 10 incorrectly classified tweets and discuss them * [2 points] c. Compare the results (cf. classification report) with the results obtained by VADER on the same tweets and discuss the differences.

1.3.4 [Optional! (will not be graded)] Question 8: trying to improve the model

- [2 points] a. Think of some ways to improve the scikit-learn Naive Bayes model by playing with the settings or applying linguistic preprocessing (e.g., by filtering on part-of-speech, or removing punctuation). Do not change the classifier but continue using the Naive Bayes classifier. Explain what the effects might be of these other settings
- [1 point] b. Apply the model with at least one new setting (train on the airline tweets using 80% training, 20% test) and generate the scores
- [1 point] c. Discuss whether the model achieved what you expected.

1.4 End of this notebook

[ ]: