# group21_lab3

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}

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

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

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

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

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}

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

### 1.2.3 Question1 Answer

First thing to note is that the scores of “neg”, “neu”, and “pos” can range from 0 to 1. The scores are actually ratios for proportions of the text that fall in each category. The scores of “neg”, “neu”, and “pos” do no take VADER’s rule based enhancement into account. The rule based enhancement are applied when calculating the compound score. The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). Standardized thresholds

for the compound score are: - Compound score > 0.05: Positive sentiment - Compound score < -0.05: Negative sentiment - Compound score between -0.05 and 0.05: Neutral sentiment

VADER also considers the intensity of the sentiment: Amplifiers enhance the sentiment of the words they modify; Punctuation and emoticons are also sentiment modifiers. When negation is applied, VADER flips the sentiment of the words the negation is applied to. Word-forms (e.g., "love", "loves", and "loved") are treated as seperate instances in the lexicon. [https://github.com/cjhutto/vaderSentiment/tree/master]

- Sentence 1
  - This output is plausible. The verb "love" results in a high positive score as it has a high positive sentiment in VADER's lexicon. Furthermore, the words "I" and "apples" are neutral words. The compound score is also reasonable as the proportions of the text mainly fall in the positive (and neutral) sentiment. Thus, having a high compound score that indicates a positive sentiment is correct.
- Sentence 2
  - This output is plausible. The use of "don't" before the word "love" results in a high negative score. This is because "love" is a strong positive word and the "don't" negates it. VADER thus flips the sentiment to a negative score. Furthermore, the words "I" and "apples" are neutral words. The compound score is also plausible. The compound score indicates a negative sentiment, which is on par with the word's used.
- Sentence 3
  - This output is plausible. The verb "love" results in a high positive score as it has a high positive sentiment in VADER's lexicon. Additionally, the smiley face at the end is also marked as a positive sentiment in the lexicon of VADER. Thus, the positive score increases. Furthermore, the words "I" and "apples" are neutral words. The compound score is also plausible. The compound scores show that the sentence has a positive sentiment. Additionally, the compound score is higher than that of sentence 1, which makes sense as the smiley face reinforces a highers positive sentiment.
- Sentence 4
  - This output is plausible. The word "ruins" is interpreted as a negative word, while the other words are neutral. We would expect the negative sentiment to be higher than the neutral sentiment, but perhaps because the sentence is a statement without any emotional value words, results in it having a higher neutral score Compound score seems plausible too, because the compound score indicates a negative sentiment.
- Sentence 5
  - This output is plausible. The word "ruins" is interpreted as a negative word and since it is negated by the "not", the sentiment is marked positive. Furthermore, the positive senitment is furhter reinforced by the word "certainly" which appears before the word "not". The other words are neutral (like in sentence 6). Compound score suggests a positive sentiment. Taking the positive scores into account of the sentence, this outcome seems plausible.
- Sentence 6
  - This output is NOT plausible. The "neg" categorization happened due to the verb "lies", which has a mean sentiment rating of -1.8 in VADER's `lexicon.txt`. However, the "lies" stems from the verb "to lay" and not "to lie". VADER should have been able to disambiguate the word based on the context. Furthermore, the sentence does not include any words that have an emotional value. Thus, it should only have a neutral score Additionally, the compound score indicates a negative sentiment. However, this is

also wrong since wrong scores were given. (A “neg” score was given. And since all the scores of “neg,”pos”, and “neu” add up to 1, a wrongly given “neg” score influences the scores of the other categories.)

- Sentence 7
  - This output is NOT plausible. In this sentence, mostly neutral words are used. The positive inclined score is due to the use of the word “like” (mean sentiment rating: 1.5), which can be seen as “to like” but also used for comparison. VADER should be able to tell the difference given the context here, meaning the “like” for comparison should have been found out. The compound score suggests that there is a positive sentiment. However, it should be a neutral sentiment. Thus, this is not correct.

#### 1.2.4 [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.

[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': 'negative', 'text_of_tweet': 'Michelle Trachtenberg has
sadly passed away at the age of 39.', 'tweet_url':
'https://x.com/DiscussingFilm/status/1894801472610611220'}
```

### 1.2.5 [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. * [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.

[4]:
```
import spacy
from nltk.sentiment.vader import SentimentIntensityAnalyzer

vader_model = SentimentIntensityAnalyzer()
nlp = spacy.load('en_core_web_sm')

def run_vader(textual_unit,
              lemmatize=False,
              parts_of_speech_to_consider=None):

    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))

    return scores
```

[5]:
```
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'
```

[6]:
```
from sklearn.metrics import classification_report

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

# Settings (to change for different experiments)
to_lemmatize = True
pos = set()

for id_, tweet_info in my_tweets.items():
    the_tweet = tweet_info['text_of_tweet']
    vader_output = run_vader(the_tweet, lemmatize=to_lemmatize)
    vader_label = vader_output_to_label(vader_output)

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

# use scikit-learn's classification report
```

```
report = classification_report(gold, all_vader_output, digits = 3)
print(report)
```

```
              precision    recall  f1-score   support

    negative      0.727     0.842     0.780        19
     neutral      0.846     0.846     0.846        13
    positive      0.800     0.667     0.727        18

    accuracy                          0.780        50
   macro avg      0.791     0.785     0.785        50
weighted avg      0.784     0.780     0.778        50
```

**Answer 3a.:** Precision, recall, and F1-score are used to measure how well VADER classified the tweets.

Precision tells us how many of the tweets that VADER labeled as a certain sentiment were actually correct. For example, VADER had a precision of 72.7% for negative tweets, meaning that when it said a tweer was negative, it was correct about 73% of the time. For neutral tweets, precision is higher, making it the most reliable category. Positive sentiment had a precision of 80%, which is similar to neutral but still leaves room for error.

Recall, on the other hand, measures how many actual tweers of each sentiment were correctly identified. VADER correctly caught 84.2% of negative tweets, which is a strong result, and 84.6% of neurtal tweets, meaning it was also good at neutral detection. However, recall for positive tweets was lower at 66.7%, meaning VADER missed quite a few of them.

The F1-score balances precision and recall, and in this casse, neutral tweets also had the heighest score at 84.6%, negative tweets followed at 78%, and positive tweets was the weakest at 72.7%.

To answer the question which score matters the most, we need to take into account what we want to acheive. If the goal is to catch as many tweets of a certain sentiment as possible, recall is the most important because it tells us how many we missed. This is particularly relevant for negative sentiment, where missing complaints or criticism could be a problem (for example, for companies looking to understand the sentiment about their products). If we want to make sure that when VADER labels somehing, its actually correct, then precision is more important. That would matter more for positive sentiment, where we want to be sure that tweets labeled as positive a truly positive. However, since F1-score balances them both, it could be generally said that its the best way to compare overall performance, and in this case, it confirms that VADER handled neutral and negative tweets well but struggled with positive ones.

[7]:
```
# --Task 3.b--

import pandas as pd
pd.set_option("display.max_colwidth", None)

misclassified_tweets = []

# Going through each tweet and comparing VADER prediction to human label
```

```
for idx, (true_label, predicted_label, text) in enumerate(zip(gold,␣
 ↪all_vader_output, tweets)):
    if true_label != predicted_label:
        misclassified_tweets.append({
            "Tweet ID": idx + 1,
            "Tweet Text": text,
            "Gold Label": true_label,
            "VADER Prediction": predicted_label
        })

df_misclassified = pd.DataFrame(misclassified_tweets)
df_positive_errors = df_misclassified[df_misclassified["Gold Label"] ==␣
 ↪"positive"].head(10)
df_negative_errors = df_misclassified[df_misclassified["Gold Label"] ==␣
 ↪"negative"].head(10)
df_neutral_errors = df_misclassified[df_misclassified["Gold Label"] ==␣
 ↪"neutral"].head(10)
```

[8]:
```
print("MISCLASSIFIED POSITIVE TWEETS: \n")
print(df_positive_errors.to_string(index=False))
```

```
MISCLASSIFIED POSITIVE TWEETS:

 Tweet ID
Tweet Text Gold Label VADER Prediction
       11
This is Brooks. He found a giant stick on his walk and stubbornly carried it all
the way home. 14/10   positive         negative
       18
Spectacular photograph, Don!   positive          neutral
       21 Good news, Elon Musk has tens of billions in loans secured against
Tesla. If Tesla goes bankrupt he will go completely bankrupt as all his current
possessions will be needed to pay his debts. Time for Tesla to tank.   positive
negative
       23                                                    If you're having
a bad day, remember that a Golden Retriever once followed a Google Maps
photographer in South Korea and ended up in a thousand photos. Good boi.
positive         negative
       45
Russia's illegal war against Ukraine is not over yet. But Canada will be there,
standing with Ukraine, until it is.   positive         negative
       46
BLUE GHOST ALMOST TO THE MOON WOWWWWW   positive         negative
```

**Answer 3.b (1):** - Tweet ID 11: The word "stubbornly" is in the lexicon with a negative score of -1.4. Since there aren't any strong positive words in the sentence that appear in the lexicon, this negative value skews the overall sentiment. Additionally, "14/10" is a highly positive rating in human interpretation, but VADER does not assign sentiment to numerical ratings, so it was

ignored. - Tweet ID 18: The word “spectacular” is not in the lexicon, so VADER doesn’t register it as positive. Since “photograph” and “Don” also aren’t in the lexicon, VADER fails to detect any positive sentiment. Additionally, punctuation amplifies sentiment, but since there is only one exclamation mark, it wasn’t enough to push the sentiment into the positive range. - Tweet ID 21: The word “good” is in the lexicon with a positive score of 1.9, which means it does contribute to the sentiment calculation. However, “news” is not in the lexicon, so “good news” as a phrase is not explicitly recognized as positive sentiment. At the same time, the words “bankrupt” (-2.6) and “debt” (-1.5) are in the lexicon with strong negative scores, and these values heavily influence the total sentiment score. Since VADER follows a summation rule, the multiple high-magnitude negative words overpower the single positive word “good” (1.9). As a result, despite starting with a seemingly positive phrase, the overall sentiment score becomes negative due to the dominance of strongly negative financial terms. - Tweet ID 23: The phrase “bad day” is commonly understood as negative, but “bad” is the only word from that phrase in the lexicon, and it has a very strong score of -2.5. Since VADER assigns more weight to high-scoring negative words, the sentiment gets pulled toward the negative side. The phrase “Good boi” is very positive to humans, but “boi” (which a social media slang), is not in the lexicon, meaning VADER does not recognize this as positive sentiment. Because the first strong word in the tweet was negative, VADER treated the tweet as overall negative. - Tweet ID 45: The words “illegal” (-2.6) and “war” (-2.9) are strongly negative in the lexicon, which means the first half of the sentence gets assigned a high negative sentiment score. However, VADER does apply a rule for contrastive conjunctions like “but”, meaning that the second part of the sentence (“Canada will be there, standing with Ukraine”) is supposed to carry more weight in the final sentiment calculation. Despite this rule, the positive words in the second half of the sentence are not strong enough to fully neutralize the very high-magnitude negative words in the first half. Since words like “Canada” and “standing with Ukraine” are not in the lexicon, they do not contribute a clear positive sentiment score. The lack of high-scoring positive words likely resulted in the earlier negative words dominating the final classification despite the contrastive conjunction rule. - Tweet ID 46: The VADER lexicon does not contain “BLUE” or “WOWWWWW”, meaning these words were ignored in the sentiment analysis. However, “WOW” (without elongation) is in the lexicon with a strong positive score of 2.8, so if the tweet had used “WOW” instead of “WOWWWWW”, it likely would have contributed a strong positive sentiment. The key reason VADER misclassified this tweet as negative is the presence of “ghost”, which is in the lexicon with a negative score of -1.3. Since no other recognized positive words were present, the negative weight from “ghost” pulled the overall sentiment score into the negative range.

[9]:
```
print("MISCLASSIFIED NEGATIVE TWEETS: \n")
print(df_negative_errors.to_string(index=False))
```

```
MISCLASSIFIED NEGATIVE TWEETS:

 Tweet ID
Tweet Text Gold Label VADER Prediction
        3
President Trump and Elon Musk are running this government like a discount
furniture store.   negative         positive
        7
JUST IN: Bitcoin falls under $84,000   negative          neutral
       40 Elon Musk rolls up to the cabinet meeting he's attending dressed like
a bum, giggling about his 'tech support' t-shirt like a 13 year-old boy. What an
```

```
embarrassment.   negative         positive
```

**Answer 3.b (2):** - Tweet ID 3: The phrase "like a discount furniture store" was likely misinterpreted because of the word "like", which has a positive sentiment score of 1.5 in the lexicon. While in context, this phrase is meant as criticism, VADER sees "like" as expressing a positive comparison, which shifts the sentiment score upwards. Since VADER applies valence shifting rules but not deep semantic understanding, it doesn't recognize the sarcastic intent behind the comparison, leading to an incorrect positive classification. - Tweet ID 7: The key issue here is that "falls" is not in the VADER lexicon. For humans, when discussing financial markets, "falls" generally carries a negative implication. Without a clear negative word in the lexicon, VADER doesn't recognize this as a negative financial event, leading to a neutral classification. However, the label we chose for this tweet could be debateable: it could be neutral if you account for the fact that some people either refer to a drop in a stock as a good thing, as it presents an opportunity to buy it at a lower price, or as a bad thing since you are technically losing money. In that case the VADER label might be correct. However we chose a negative sentiment here since we believe it to be the most common interpatation, as usually when a stock price falls it isnt a positive event.
- Tweet ID 40: The word "giggling" has a positive score of 1.5, and "support" has an even stronger positive score of 1.7, which likely increased the overall sentiment score. Meanwhile, "embarrassment" is correctly listed in the lexicon as negative (-1.9), but it wasn't enough to outweigh the positive words. While a human can easily tell that calling someone a "bum" (which isnt included in the lexicon) and comparing them to a "13-year-old boy" is mocking, VADER sees "giggling" and "support" as positive and doesn't recognize the insulting intent behind the phrase.

[10]:
```
print("MISCLASSIFIED NEUTRAL TWEETS: \n")
print(df_neutral_errors.to_string(index=False))
```

```
MISCLASSIFIED NEUTRAL TWEETS:

 Tweet ID
Tweet Text Gold Label VADER Prediction
       29                                  CGI primates from "Better Man" and
"Kingdom of the Planet of the Apes" face off in this year's Oscars race for best
visual effects.    neutral         positive
       33 Artists including Kate Bush and Cat Stevens made an album of white
noise in empty studios, protesting a U.K. proposal to give AI firms access to
copyrighted music.    neutral         negative
```

**Answer 3.b (3):** - Tweet ID 29: The tweet is simply reporting facts about the Oscars, so humans interpret it as neutral, but VADER assigns positive sentiment due to two words in its lexicon: "Better" (1.9) contributes positivity, "Best" (3.2) has an even stronger positive weight. Since VADER is a summation-based model, these two words alone likely pushed the compound score into the positive range, even though the tweet itself does not express an opinion. Additionally, VADER does not understand that "best" is referring to an awards category rather than an evaluative sentiment. This is a common issue with domain-specific words (like award categories) being misinterpreted as subjective sentiment. - Tweet ID 33: Two words contributed to the misclassification as negative: "Protest" (-1.0) has a negative sentiment in the lexicon, even though in this case, it's being used in a neutral factual sense, and "Empty" (-0.8) that also contributes negativity, even though it refers to a physical space rather than a negative emotional tone.

### 1.2.6 [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.
- – Are all parts of speech equally important for sentiment analysis? Explain why or why not.

**Answers to Question 4** First, we unzip the `airlinetweets.zip` file.

[11]:
```
import os
import zipfile

# Unzip the 'airlinetweets.zip' file if not already unzipped
extract_folder = 'airlinetweets'

if not os.path.exists(extract_folder):
    zip_file_path = 'airlinetweets.zip'

    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        zip_ref.extractall()
```

The 'airlinetweets' folder in the `airlinetweets.zip` file contains three folders: 'negative', 'neutral', and 'positive'.

Each of these three folders (representing gold labels) contains hundreds of .txt files, each file containing the raw text of an individual tweet.

E.g., "@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 " is the content of a .txt file in the

'negative' folder, as its sentiment is negative.

Now, we define a function to run for each experiment above.

[12]:
```
import pandas as pd

# Define function to run for each experiment
def vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
):
    # Define the paths to the sentiment folders
    sentiment_folders = ['negative', 'neutral', 'positive']  # gold labels
    results = []

    # Iterate through each sentiment folder and process the tweets
    for sentiment in sentiment_folders:
        folder_path = os.path.join(extract_folder, sentiment)

        for filename in os.listdir(folder_path):
            if filename.endswith('.txt'):
                file_path = os.path.join(folder_path, filename)

                # Read the tweet from the file
                with open(file_path, 'r', encoding='utf-8') as file:
                    tweet = file.read().strip()

                    # Run VADER on the tweet
                    scores = run_vader(tweet, lemmatize,␣
 ↪parts_of_speech_to_consider)
                    vader_label = vader_output_to_label(scores)

                    # Store the results
                    results.append({
                        'tweet': tweet,
                        'sentiment': sentiment,  # GOLD: from folder name
                        'scores': scores,  # from VADER
                        'label': vader_label
                    })

    # Create a DataFrame from the results
    results_df = pd.DataFrame(results)

    # Prepare the gold labels and VADER predictions for evaluation
    gold_labels = results_df['sentiment'].tolist()
    vader_predictions = results_df['label'].tolist()

    # Generate the classification report
```

```
    report = classification_report(gold_labels, vader_predictions, digits=3)

    print(report)
```

Finally, we perform the experiments. * Experiment 1: Run VADER (as it is) on the set of airline tweets

[13]:
```
# Settings (to change for different experiments)
to_lemmatize = False
pos = None

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.797     0.515     0.625      1750
     neutral      0.605     0.506     0.551      1515
    positive      0.559     0.884     0.685      1490

    accuracy                          0.628      4755
   macro avg      0.654     0.635     0.621      4755
weighted avg      0.661     0.628     0.620      4755
```

- Experiment 2: Run VADER on the set of airline tweets after having lemmatized the text

[14]:
```
# Settings (to change for different experiments)
to_lemmatize = True
pos = None

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.786     0.522     0.628      1750
     neutral      0.598     0.488     0.538      1515
    positive      0.557     0.881     0.682      1490

    accuracy                          0.624      4755
   macro avg      0.647     0.630     0.616      4755
weighted avg      0.654     0.624     0.616      4755
```

- Experiment 3: Run VADER on the set of airline tweets with only adjectives

[15]:
```
# Settings (to change for different experiments)
to_lemmatize = False
pos = {'ADJ'}

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.870     0.210     0.339      1750
     neutral      0.403     0.892     0.555      1515
    positive      0.665     0.438     0.528      1490

    accuracy                          0.499      4755
   macro avg      0.646     0.513     0.474      4755
weighted avg      0.657     0.499     0.467      4755
```

- Experiment 4: Run VADER on the set of airline tweets with only adjectives and after having lemmatized the text

[16]:
```
# Settings (to change for different experiments)
to_lemmatize = True
pos = {'ADJ'}

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.868     0.210     0.339      1750
     neutral      0.403     0.892     0.556      1515
    positive      0.664     0.438     0.528      1490

    accuracy                          0.499      4755
   macro avg      0.645     0.513     0.474      4755
weighted avg      0.656     0.499     0.467      4755
```

- Experiment 5: Run VADER on the set of airline tweets with only nouns

[17]:
```
# Settings (to change for different experiments)
to_lemmatize = False
pos = {'NOUN'}

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.730     0.143     0.240      1750
     neutral      0.358     0.817     0.498      1515
    positive      0.532     0.340     0.415      1490

    accuracy                          0.420      4755
   macro avg      0.540     0.433     0.384      4755
weighted avg      0.549     0.420     0.377      4755
```

- Experiment 6: Run VADER on the set of airline tweets with only nouns and after having lemmatized the text

[18]:
```
# Settings (to change for different experiments)
to_lemmatize = True
pos = {'NOUN'}

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.715     0.157     0.257      1750
     neutral      0.358     0.809     0.496      1515
    positive      0.521     0.331     0.405      1490

    accuracy                          0.419      4755
   macro avg      0.531     0.432     0.386      4755
weighted avg      0.540     0.419     0.379      4755
```

- Experiment 7: Run VADER on the set of airline tweets with only verbs

[19]:
```
# Settings (to change for different experiments)
to_lemmatize = False
```

```
pos = {'VERB'}

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.774     0.288     0.420      1750
     neutral      0.383     0.810     0.520      1515
    positive      0.568     0.343     0.428      1490

    accuracy                          0.472      4755
   macro avg      0.575     0.480     0.456      4755
weighted avg      0.585     0.472     0.454      4755
```

- Experiment 8: Run VADER on the set of airline tweets with only verbs and after having lemmatized the text

[20]:
```
# Settings (to change for different experiments)
to_lemmatize = True
pos = {'VERB'}

# Run experiment
vader_experiment(
        lemmatize=to_lemmatize,
        parts_of_speech_to_consider=pos
)
```

```
              precision    recall  f1-score   support

    negative      0.741     0.295     0.422      1750
     neutral      0.377     0.780     0.508      1515
    positive      0.568     0.352     0.434      1490

    accuracy                          0.468      4755
   macro avg      0.562     0.476     0.455      4755
weighted avg      0.571     0.468     0.454      4755
```

**Answers to 4b: Comparing the scores** - *Q: Does lemmatisation help?* A: First, we compare experiment 1 (no lemmatisation, no PoS filter) with experiment 2 (lemmatisation, no PoS filter). The precision, recall, and f1-score are roughly the same or slightly higher for each sentiment individually (negative, neutral, and positive) as well as macro-averaged for experiment 1. Thus, for this experiment lemmatisation does not provide a substantial benefit. Since the original forms of words are well represented in VADER's lexicon.txt, lemmatisation does not yield an advantage. We observe

similar results when comparing experiment 3 (no lemmatisation, only adjectives) with experiment 4 (lemmatisation, only adjectives), experiment 5 (no lemmatisation, only nouns) with experiment 6 (lemmatisation, only nouns), and experiment 7 (no lemmatisation, only verbs) with experiment 8 (lemmatisation, only verbs). - *Q: Are all parts of speech equally important for sentiment analysis?* A: Now, we compare the macro averages of experiments 3 (no lemmatisation, only adjectives), 5 (no lemmatisation, only nouns), and 7 (no lemmatisation, only verbs) with eachother, as lemmatisation does not help and the macro-averaged precision, recall, and f1-score of experiment 1 are higher than the scores for the three aforementioned experiments. This suggests that sentiment analysis works best when considering all parts-of-speech. We first compare the macro-averaged f1-scores, since they provide a single metric balancing both precision and recall. We observe the highest macro-averaged f1-score (of experiments 3, 5, and 7) in experiment 3, suggesting that adjectives play a crucial role in sentiment analysis. People often use adjectives to provide descriptive context about other parts-of-speech. Therefore, adjectives are often rich in sentiment. Adjectives are followed closely by verbs, with nouns clearly in last place (again, according to macro-averaged f1-score). As seen in task 1, verbs such as "love" or "like" carry significant sentiment ratings. However, this sentiment is heavily dependent on context (e.g., the verb "like" can be negative, neutral, or positive, depending on the surrounding words). Lastly, nouns typically serve as subjects or objects in sentences and may not inherently express sentiment. While providing context, they lack the emotional quality that is possessed by adjectives and verbs.

## 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?
  - does the frequency threshold affect the scores? Why or why not according to you?

[21]:
```
import os
import zipfile
import pathlib
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_files
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import classification_report
```

```
from sklearn.naive_bayes import MultinomialNB
# To ignore the UserWarnings
import warnings
warnings.simplefilter("ignore", UserWarning)

# Unzip the 'airlinetweets.zip' file if not already unzipped
extract_folder = 'airlinetweets'

if not os.path.exists(extract_folder):
    zip_file_path = 'airlinetweets.zip'

    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        zip_ref.extractall()

# Load the airline tweets dataset
cwd = pathlib.Path.cwd()
airline_tweets_folder = cwd.joinpath('airlinetweets')
path = str(airline_tweets_folder)
airline_tweets = load_files(path)

# Define the min_df values to test
min_df_values = [2,5,10]
for i in min_df_values:
    # initialize airline object, and then turn airline tweets train data into a␣
 ↪vector
    airline_vec = CountVectorizer(min_df=i, # If a token appears fewer times␣
 ↪than this, across all documents, it will be ignored
                                  tokenizer=nltk.word_tokenize, # we use the nltk␣
 ↪tokenizer
                                  stop_words=stopwords.words('english')) #␣
 ↪stopwords are removed

    # Bag of words representation of the airline tweets
    airline_counts = airline_vec.fit_transform(airline_tweets.data)
    # Split dataset into bag of words training and testing sets (80-20 split)
    Xbag_train, Xbag_test, ybag_train, ybag_test = train_test_split(
        airline_counts,
        airline_tweets.target,
        test_size = 0.20,
        random_state=21  # we set the seed
        )
    # Train a Multinomial Naive Bayes classifier
    clf = MultinomialNB().fit(Xbag_train, ybag_train)
    # Predict the train results
    ybag_pred = clf.predict(Xbag_test)
    print(f"Classification report for min_df={i}")
    print("Bag of Words Representation")
```

```
    print(classification_report(ybag_test, ybag_pred,␣
↪target_names=airline_tweets.target_names))

    # TF-IDF Representation
    # Convert raw frequency counts into TF-IDF values
    tfidf_transformer = TfidfTransformer()
    airline_tfidf = tfidf_transformer.fit_transform(airline_counts)
    # Split dataset into TD-IDF training and testing sets (80-20 split)
    Xtf_train, Xtf_test, ytf_train, ytf_test = train_test_split(
        airline_tfidf,
        airline_tweets.target,
        test_size = 0.20,
        random_state=21
        )
    # Train a Multinomial Naive Bayes classifier
    clf = MultinomialNB().fit(Xtf_train, ytf_train)
    # Predict the train results
    ytf_pred = clf.predict(Xtf_test)
    print("TF-IDF Representation")
    print(classification_report(ytf_test, ytf_pred, target_names=airline_tweets.
↪target_names))
```

```
Classification report for min_df=2
Bag of Words Representation
              precision    recall  f1-score   support

    negative       0.79      0.90      0.84       325
     neutral       0.84      0.71      0.77       311
    positive       0.84      0.85      0.84       315

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

TF-IDF Representation
              precision    recall  f1-score   support

    negative       0.77      0.89      0.83       325
     neutral       0.83      0.69      0.76       311
    positive       0.84      0.84      0.84       315

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

Classification report for min_df=5
Bag of Words Representation
              precision    recall  f1-score   support
```

```
    negative       0.80      0.88      0.84       325
     neutral       0.81      0.75      0.78       311
    positive       0.85      0.83      0.84       315

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

TF-IDF Representation
              precision    recall  f1-score   support

    negative       0.77      0.87      0.82       325
     neutral       0.80      0.73      0.76       311
    positive       0.85      0.82      0.83       315

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

Classification report for min_df=10
Bag of Words Representation
              precision    recall  f1-score   support

    negative       0.80      0.87      0.83       325
     neutral       0.80      0.77      0.79       311
    positive       0.87      0.82      0.84       315

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

TF-IDF Representation
              precision    recall  f1-score   support

    negative       0.79      0.85      0.82       325
     neutral       0.78      0.76      0.77       311
    positive       0.86      0.81      0.83       315

    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 Question 5b

To evaluate the experiment, the F1-score will be primarily used, as it balances both recall and precision, providing a comprehensive measure of the model's performance. Interestingly, when

averaging the F1-scores for each representation (TF-IDF and bag of words) and each frequency threshold (2, 5 and 10), the bag of words approach performs slightly better than TF-IDF. This could be because TF-IDF gives more weight to unique words, which may cause it to overlook common but important words in sentiment classification. In contrast, bag of words treats all words equally. This results in the bag of words approach capturing sentiment related words more effectively. Furthermore, when comparing F1-scores, bag of words clearly outperforms TF-IDF in the neutral category. For the negative and positive categories, bag of words either slightly outperforms TF-IDF or achieves the same F1-score. This suggests that bag of words captures essential sentiment related words more effectively than TF-IDF. Notably, the negative and positive categories perform better than the neutral category in all experiments. This suggests that the model is better at distinguishing positive and negative sentiments than neutral ones. The positive category consistently scores the highest F1-score, followed closely by the negative category. This could be because postive and negative sentiments have sentiment indicating words, something the neutral sentiment lacks.

The frequency threshold was another part of this experiment. A higher frequency threshold ignores unique words (words that appear infrequently). However, it does not seem to have a significant effect in this case. For both TF-IDF and bag of words, a frequency threshold of 2 gives the best performance. This suggests that filtering out extremely rare words (words that appear less than 2 times) while still retaining moderately infrequent ones (words that appear at least twice) provides the best balance for sentiment classification in this case. If anything, increasing the threshold reduces noise, but it does not lead to a substantial improvement in performance. Thus, the frequency threshold has a minor impact on the scores. However, it seems like a higher frequency has a small positive impact on the neutral category, possibly by eliminating rare words that add confusion rather than clarity.

### 1.3.3 [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?
  - [1 point] Which features did you not expect and why ?
  - [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?

[27]:
```
airline_vec = CountVectorizer(
    min_df=2,
    tokenizer=nltk.word_tokenize,
    stop_words=stopwords.words('english')
)

# bow matrix
airline_count = airline_vec.fit_transform(airline_tweets.data)
```

```
docs_train, docs_test, y_train, y_test = train_test_split(
    airline_count,
    airline_tweets.target, # the category values for each tweet
    test_size = 0.20, # we use 80% for training and 20% for testing
    random_state=21
    )

clf = MultinomialNB().fit(docs_train, y_train)

y_pred = clf.predict(docs_test)

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 1529.0 @
0 1402.0 united
0 1247.0 .
0 424.0 ``
0 398.0 ?
0 386.0 flight
0 365.0 !
0 328.0 #
0 216.0 n't
0 154.0 ''
0 132.0 's
```

```
0 118.0 service
0 110.0 virginamerica
0 110.0 :
0 98.0 get
0 96.0 customer
0 95.0 delayed
0 90.0 cancelled
0 84.0 bag
0 81.0 time
0 81.0 plane
0 79.0 hours
0 74.0 ;
0 71.0 -
0 70.0 …
0 69.0 gate
0 68.0 'm
0 68.0 &
0 67.0 http
0 66.0 still
0 64.0 late
0 63.0 airline
0 62.0 would
0 61.0 2
0 60.0 help
0 59.0 hour
0 59.0 amp
0 53.0 flights
0 52.0 ca
0 50.0 like
0 49.0 delay
0 49.0 $
0 48.0 waiting
0 48.0 one
0 47.0 never
0 45.0 worst
0 45.0 've
0 44.0 luggage
0 44.0 (
0 42.0 flightled
0 42.0 due
0 41.0 u
0 40.0 us
0 40.0 fly
0 40.0 )
0 38.0 day
0 38.0 check
0 38.0 back
0 37.0 wait
```

```
0 36.0 thanks
0 36.0 really
0 35.0 another
0 34.0 seat
0 34.0 people
0 34.0 ever
0 34.0 even
0 34.0 bags
0 33.0 3
0 32.0 lost
0 32.0 last
0 32.0 hold
0 31.0 staff
0 31.0 guys
0 31.0 going
0 31.0 baggage
0 30.0 crew
0 29.0 ticket
0 29.0 days
0 28.0 terrible
0 28.0 seats
----------------------------------------
Important words in neutral documents
1 1400.0 @
1 522.0 ?
1 488.0 .
1 307.0 jetblue
1 282.0 :
1 263.0 southwestair
1 255.0 united
1 252.0 #
1 239.0 ``
1 219.0 flight
1 182.0 americanair
1 177.0 http
1 165.0 !
1 154.0 usairways
1 127.0 's
1 78.0 virginamerica
1 75.0 get
1 75.0 -
1 69.0 ''
1 65.0 )
1 64.0 flights
1 58.0 help
1 57.0 please
1 57.0 (
1 54.0 need
```

1 51.0 n't
1 45.0 dm
1 44.0 ;
1 43.0 …
1 41.0 tomorrow
1 38.0 us
1 38.0 know
1 38.0 &
1 37.0 would
1 37.0 flying
1 37.0 fleet
1 37.0 fleek
1 34.0 thanks
1 34.0 'm
1 33.0 way
1 32.0 "
1 32.0 hi
1 32.0 change
1 32.0 amp
1 31.0 "
1 31.0 cancelled
1 30.0 one
1 29.0 like
1 29.0 fly
1 29.0 could
1 27.0 number
1 27.0 airport
1 26.0 go
1 25.0 today
1 23.0 time
1 23.0 new
1 22.0 see
1 21.0 use
1 21.0 travel
1 21.0 destinationdragons
1 20.0 check
1 20.0 add
1 19.0 want
1 19.0 start
1 19.0 question
1 19.0 make
1 19.0 chance
1 18.0 trying
1 18.0 ticket
1 18.0 sent
1 18.0 rt
1 18.0 last
1 18.0 guys

```
1 18.0 follow
1 18.0 dfw
1 17.0 weather
1 17.0 reservation
1 17.0 gate
1 17.0 flightled
1 17.0 first
----------------------------------------
Important words in positive documents
2 1305.0 @
2 1052.0 !
2 756.0 .
2 316.0 #
2 311.0 southwestair
2 280.0 thanks
2 275.0 jetblue
2 260.0 thank
2 248.0 united
2 234.0 ``
2 184.0 flight
2 171.0 :
2 166.0 americanair
2 138.0 great
2 131.0 usairways
2 90.0 )
2 89.0 service
2 87.0 virginamerica
2 77.0 http
2 66.0 customer
2 64.0 best
2 63.0 much
2 63.0 love
2 62.0 guys
2 60.0 ;
2 58.0 's
2 56.0 awesome
2 51.0 -
2 48.0 good
2 45.0 time
2 45.0 got
2 44.0 airline
2 43.0 &
2 41.0 amazing
2 40.0 crew
2 39.0 today
2 38.0 us
2 38.0 get
2 36.0 amp
```

```
2 33.0 n't
2 32.0 …
2 31.0 help
2 30.0 made
2 30.0 ''
2 29.0 fly
2 28.0 home
2 28.0 gate
2 28.0 flying
2 27.0 see
2 27.0 day
2 27.0 appreciate
2 27.0 'm
2 26.0 response
2 26.0 new
2 26.0 back
2 25.0 work
2 25.0 tonight
2 25.0 flights
2 25.0 're
2 24.0 well
2 24.0 nice
2 23.0 ever
2 23.0 ?
2 22.0 (
2 22.0 'll
2 21.0 like
2 21.0 know
2 21.0 helpful
2 21.0 always
2 20.0 team
2 20.0 first
2 19.0 u
2 19.0 staff
2 19.0 job
2 19.0 attendant
2 18.0 yes
2 18.0 please
2 18.0 plane
2 18.0 one
2 18.0 happy
```

**Answers to Questions for 6B**

- Which features did you expect for each separate class and why?

Negative - I would expect “cancelled”, “delayed”, “late”, “terrible” because they have negative connotations that could easily classify them as a negative tweet. Additionally “terrible” or “worst” because those are negatively emotionally charged.

Positive - I would expect “love,”best”, “great” because these words could easily be classified for positive, due to them being positively emotionally charged. however there is potential for sarcasm that could miscategorize these I suppose

Neutral - nothing emotionally charged in particular, I would imagine features such as the name of the airlines, “jetblue”;“united”;“virgin america”, etc. would frequently show up as neutral tweets would likely be more factual and less opinioned. Neutral tweets seem to have many features such as “help” “please” “thanks” and then specific words that indicate that they were likely customer support questions.

- Which features did you not expect and why?

Honestly I didn’t expect so much of the punctuation to be counted as relevant features. Additionally, the frequency of words in negative tweets don’t have that many words that can particularly be attributed to having a negative connotation at face value – “terrible” is ranked towards the bottom, “lost” and “worst” also are ranked a lot lower than features such as “delayed” and “cancelled”. To get to a positive connotation word in the positive tweets, you only need to count from rank 14 down and the obviously positive connotation features are grouped closer together, whereas in the negative tweets the obviously negative connotation words are spread out further apart. Looking at the features altogether with context knowing theyre negative tweets makes the complaints obvious, but if you’re looking at them as isolated features the words with positive connotation appear a lot more frequently in positive tweets that make it more obvious it is a positive tweet.

- 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?

Removing features such as ’ ’, &, :, n’t, ’s, and those various symbols would be a first. I don’t think those provide any meaning that can give us information on the positive, negative or neutral nature of the text. I would keep “?” and “!” because those could imply emotion (though they can be used both positively and negatively. Additionally, neutral tweets seem to appear more “factual” in the sense that there are less emotionally charged words in there, so many neutral tweets could potentially be non-aggressive questions towards customer support or towards the general twitterbase). I remove the @ symbol as well – while this symbol implies it is trying to get the attention of a particular user/airline, it is the most frequent feature throughout pos,neg and neutral tweets thereby not providing any useful information that can indicate to us how to categorize the tweets any better than if we had left them in. Since it doesn’t seem like anyone is tweeting about airlines into the void. Removing the names of airlines I would potentially do as well since they all appear towards the top of the list for pos, neg and neutral and that doesn’t provide us with new context either.

I would keep the emotionally charged words or words with negative or positive connotation such as “delay”, “cancel”, “great”, “terrible” and so on because those are the most descriptive regarding how to categorize tweets. I would keep the hastag as well potentially since it ranks fourth in frequency amongst positive tweets but 8th for negative and neutral. Perhaps it has potential

#### 1.3.4 [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.

#### 1.3.5 [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

[ ]: