Lab3 - Assignment Sentiment

This notebook describes the LAB-2 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 F_1)
- 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 F_1) 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.

Credits

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

Part I: VADER assignments

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)

[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. Vader correctly identifies the sentence as being highly positive, due to the positive word 'love' being used, and thus gives it a high positive score and a positive compound score.
2. Vader sees the negation of 'love' accurately, and thus correctly gives it a negative compound score due to 'don't love' being used in the sentence.
3. Vader seems to be able to identify the smiley in the sentence, and also sees the word 'love' being used. Thus the sentence correctly gets a positive compound score, which is even more positive than the first sentence due to the smiley.
4. Vader sees the word 'ruins' and correctly identifies this as a negative word, with a high negative score. The negative compound score also reflects the negative sentiment of the sentence accurately.
5. Vader accurately assesses the sentiment of the sentence, because the negative word 'ruins' is negated within this sentence with 'certainly not', leading to a positive compound score. The score is also not too high because this sentence is only negating a negative assessment of houses.
6. Vader is slightly off with this sentence, probably due to the word 'lies' being used in the sentence, which has multiple meanings, but the negative meaning is not used within this sentence. Vader correctly sees a lot of neutrality in this sentence, but also sees some negativity in the sentence, resulting in the compound score being off. The compound score reflects a pretty negative sentence, whereas this sentence is more of a neutral/ lightly positive sentence.
7. Vader is also slightly off in the last sentence. The sentence can be both a neutral or negative assessment of a house, because the house doesn't feel distinct from other houses. Vader correctly detects some neutrality in the sentence, due to 'any house' being a mostly neutral description. However, vader also saw the word 'likes', which probably increased the positivity score of the sentence, leading to a somewhat positive compound score. The positive and negative score should be interchanged, and the compound score should be closer to 0 or slightly negative to accurately reflect the slightly negative tone of this sentence.

[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.

We will store the tweets in the file **my_tweets.json** (use a text editor to edit). For each tweet, you should insert:

In [ ]:
```
import json
```

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

- the text of the tweet

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

```
1 {'sentiment_label': 'negative', 'text_of_tweet': 'Iâ€™m actually like so angry abo
ut Iceland"I can't stop thinking about it. ', 'tweet_url': 'https://twitter.com/Ma
xxyRainbow/status/1764168102571360681'}
```

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

[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.

a. Looking at the quantitative analysis, we can see that in all three metrics (precision, recall and F1-score) neutral is more often incorrectly classified as compared to the other two categories. This can be explained by the fact that even though a text may be neutral, it can still contain words that have a negative or positive connotation when used in a subjective text. Think with this for example of a factual statement about hurricanes: the text "a hurricane is a type of natural disaster" is undoubtedly neutral. However, the word "disaster" has a negative pretense, and thus the sentence might be classified as negative. Assuming we are trying to assess the negativity/positivity of the platform, we would consider F1-score to be the most important metric in that regard. This metric most accurately encompasses the goal of trying to get an analysis as close to reality as possible. If for example, we were trying to make automatic moderation automatically banning tweets that are too negative, precision would be more important, since you dont want to ban any innocent people.

b. neutral: words like disaster, harsh, prison and strange are interpreted as negative words, while in this context they are not used this way. The same goes in the positive direction for words like safety and security can be seen as positive, while in this specific sentence they are used very factually. This problem was described earlier in 3a.

negative: For some sentences, even though the sentence is ultimately negative, there is a lot of neutral/lightly positive words mixed in there as well (maybe in a contradiction, or someone hopes for better, or simply in their wording), making the final label different from the actual one. VADER of course cannot pick up on the subtle nuances and contextual clues (something like sarcasm or requiring knowledge about the world) very well, making sentences like sentence #6 hard to interpret. another possibility is VADER misinterpreting the sentence and using the wrong version of a word (like has multiple meanings) for its sentiment analysis. This leads it to incorrectly classifying sentences. Then there are also the sentences that dont contain any positively or negatively connotated words, thus leading to a false label. This is thus simply a lack of completeness in the VADER lexicon. For the sentence about burger king we dont really know why it is classified as positive, since we can only find either a neutral interpretation or negatively annotated words. Still the sentence is classified as lightly positive.

positive: In some sentences, some very positive words are not part of the VADER lexicon (incredible being one). Sometimes the lemmatization changes positive words into negative ones (cutest -> cut). In other sentences there is not really one clear label (the madonna tweet was meant ironic; They describe a negative thing with a positive intention). And finally, just like for the negative tweets, sometimes a sentence contains words that on their own are interpreted as the other direction -> mad is very negative, thus the whole sentence is interpreted as negative

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

vader_model = SentimentIntensityAnalyzer()
nlp = spacy.load("en_core_web_sm") # 'en_core_web_sm'
```

In [ ]:
```
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}
assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.01
assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': -0.0
```

In [ ]:
```
def run_vader(textual_unit,
              lemmatize=False,
              parts_of_speech_to_consider=None,):
    """
    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 sentence in doc.sents:
        for token in sentence:
            to_add = token.text

            if lemmatize:
                to_add = token.lemma_ if token.lemma_ != '-PRON-' else token.text

            if not parts_of_speech_to_consider:
                input_to_vader.append(to_add)
                continue
            if token.pos_ in parts_of_speech_to_consider:
                    input_to_vader.append(to_add)

    return vader_model.polarity_scores(' '.join(input_to_vader))
```

In [ ]:
```
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, to_lemmatize, pos)
    vader_label = vader_output_to_label(vader_output)

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

    if vader_label != tweet_info['sentiment_label']:
        print(f"LABEL: {tweet_info['sentiment_label']} V-LABEL: {vader_label} TWEET

# use scikit-learn's classification report
from sklearn.metrics import classification_report
print(classification_report(y_true=gold, y_pred=all_vader_output))
```

```
LABEL: negative V-LABEL: positive TWEET: ffs i did not want to wake up to this israe
l news maybe ive been too naive but i really did trust the EBU would actually do som
ething
LABEL: neutral V-LABEL: negative TWEET: Designed to withstand the harsh environment
of Mars, these squishy robots could transform how first responders determine their a
pproach to a disaster scene here on Earth. Learn how  @NASASpinoff tech could help d
isaster response: https://go.nasa.gov/3IpybN2
LABEL: positive V-LABEL: neutral TWEET: Incredible pan across Marsâ€™ surface from N
ASAâ€™s Perseverance rover ðŸ˜²
LABEL: negative V-LABEL: positive TWEET: This is what Mali, Africa looks like today.
This is what the future of Western countries looks like when you keep importing the
third world. You become the third world.
LABEL: negative V-LABEL: neutral TWEET: UK just threw a guy into jail for 2 years fo
r stickers Stickers
LABEL: negative V-LABEL: positive TWEET: Something how Dr. Fauci is revered by the L
ameStream Media as such a great professional, having done, they say, such an incredi
ble job, yet he works for me and the Trump Administration, and I am in no way given
any credit for my work. Gee, could this just be more Fake News?
LABEL: negative V-LABEL: neutral TWEET: The biggest lie the legacy media makes is na
rrative: choosing what to write about and what not to write about
LABEL: positive V-LABEL: negative TWEET: Red Pandas are the cutest creatures on eart
h
LABEL: positive V-LABEL: negative TWEET: 35 years ago madonna released â€˜like a pra
yerâ€™ and the vatican banned it, iconic.
LABEL: positive V-LABEL: neutral TWEET: my brother got my name tatted with a lil hea
rt next to it ðŸ¥²ðŸ¥²
LABEL: neutral V-LABEL: positive TWEET: Testing the safety & security on an inflatab
le car storage by WhistlinDiesel. I mean, could they have thrown anything else?
LABEL: neutral V-LABEL: negative TWEET: The US threw a dude in prison over a meme  W
e live in strange times
LABEL: negative V-LABEL: positive TWEET: This is a receipt from Burger King in 1986.
3 whoppers, 2 fries and 2 large vanilla ice cream for $8.39         Never forget wha
t they took from us ðŸ˜°
LABEL: positive V-LABEL: negative TWEET: I donâ€™t have beef with a single soul . If
ur mad get well soon
LABEL: negative V-LABEL: positive TWEET: It's laughable that anyone, male or female,
that looks like this is permitted to make health decisions for a country.
              precision    recall  f1-score   support

    negative       0.71      0.63      0.67        19
     neutral       0.43      0.50      0.46         6
    positive       0.77      0.80      0.78        25

    accuracy                           0.70        50
   macro avg       0.63      0.64      0.64        50
weighted avg       0.70      0.70      0.70        50
```

[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 F_1 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.

b.

- Lemmatisation does not seem to have a significant effect on the outcome of the algorithms results. This is probably due to the fact that most lemmatized words dont change sentiment from their original form. In the end, the judgement stays relatively the same (loving -> love doesnt suddenly become negative)
- As far as we can tell, the order from most important to least goes as follows: Adjectives -> Verbs -> Nouns. This is most likely due to the fact that adjectives are used to describe something, and depending on the choice of specific adjective you can choose what sentiment to convey (bad apple vs good apple). The noun is simply the subject of the sentence, and whether you have a positive or a bad opinion doesnt change the fact that you use them. As for verbs, there is a little bit of variation you can make depending on sentiment, but oftentimes the verb just describes a certain action, which would be described anyways as well. Some verbs can however have a sentiment of themselves (loving, hating, killing, etc.)

In [ ]:
```
import pathlib
from sklearn.datasets import load_files

cwd = pathlib.Path.cwd()
airline_tweets_folder = cwd.joinpath('airlinetweets')
print('path:', airline_tweets_folder)
print('this will print True if the folder exists:',
    airline_tweets_folder.exists())

# loading all files as training data.
str(airline_tweets_folder)
airline_tweets_train = load_files(str(airline_tweets_folder))
```

```
path: d:\school\TUDelft\Python\Artificial Intelligence\Text Mining\ba-text-mining-gr
oup19\lab_sessions\lab3\airlinetweets
this will print True if the folder exists: True
```

In [ ]:
```
print(len(airline_tweets_train.data))
```

```
4755
```

In [ ]:
```
#read the zip file "airlinetweets.zip" and extract the file "airline_tweets.json

def run_vader_on_airline_tweets(to_lemmatize: bool, pos: set):
    gold = []
    all_vader_output = []
    for tweet in airline_tweets_train.data:
        tweet = tweet.decode('utf-8')
        vader_output = run_vader(tweet, to_lemmatize, pos)
        vader_label = vader_output_to_label(vader_output)

        all_vader_output.append(vader_label)

    for target in airline_tweets_train.target:
        gold.append(airline_tweets_train.target_names[target])

    #print the classification report
    print(classification_report(y_true=gold, y_pred=all_vader_output))
```

In [ ]:
```
#run vader on airline tweets
print('As is')
run_vader_on_airline_tweets(to_lemmatize=False, pos=None)
print("_____
```

```
As is
              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


_____
__
```

In [ ]:
```
# 2. lemmatized
print('lemmatized')
run_vader_on_airline_tweets(to_lemmatize=True, pos=None)
print("_____
```

```
lemmatized
              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


_____
__
```

In [ ]:
```
# 3. adjectives
print("Only adjectives")
run_vader_on_airline_tweets(to_lemmatize=False, pos={'ADJ'})
print("_____
```

```
Only adjectives
              precision    recall  f1-score   support

    negative       0.87      0.21      0.34      1750
     neutral       0.40      0.89      0.56      1515
    positive       0.66      0.44      0.53      1490

    accuracy                           0.50      4755
   macro avg       0.65      0.51      0.47      4755
weighted avg       0.66      0.50      0.47      4755


_____
__
```

In [ ]:
```
# 4. adjectives & lemmatized
print("Only adjectives & lemmatized")
run_vader_on_airline_tweets(to_lemmatize=True, pos={'ADJ'})
print("_____
```

```
Only adjectives & lemmatized
              precision    recall  f1-score   support

    negative       0.87      0.21      0.34      1750
     neutral       0.40      0.89      0.56      1515
    positive       0.66      0.44      0.53      1490

    accuracy                           0.50      4755
   macro avg       0.65      0.51      0.47      4755
weighted avg       0.66      0.50      0.47      4755


_____
__
```

In [ ]:
```
# 5. nouns
print("Only nouns")
run_vader_on_airline_tweets(to_lemmatize=False, pos={'NOUN'})
print("_____
```

```
Only nouns
              precision    recall  f1-score   support

    negative       0.73      0.14      0.24      1750
     neutral       0.36      0.82      0.50      1515
    positive       0.53      0.34      0.41      1490

    accuracy                           0.42      4755
   macro avg       0.54      0.43      0.38      4755
weighted avg       0.55      0.42      0.38      4755


_____
__
```

In [ ]:
```
# 6. nouns & lemmatized
print("Only nouns & lemmatized")
run_vader_on_airline_tweets(to_lemmatize=True, pos={'NOUN'})
print("_____
```

```
Only nouns & lemmatized
              precision    recall  f1-score   support

    negative       0.72      0.16      0.26      1750
     neutral       0.36      0.81      0.50      1515
    positive       0.52      0.33      0.40      1490

    accuracy                           0.42      4755
   macro avg       0.53      0.43      0.39      4755
weighted avg       0.54      0.42      0.38      4755


_____
__
```

In [ ]:
```
# 7. verbs
print("Only verbs")
run_vader_on_airline_tweets(to_lemmatize=False, pos={'VERB'})
print("_____
```

```
Only verbs
              precision    recall  f1-score   support

    negative       0.77      0.29      0.42      1750
     neutral       0.38      0.81      0.52      1515
    positive       0.57      0.34      0.43      1490

    accuracy                           0.47      4755
   macro avg       0.58      0.48      0.46      4755
weighted avg       0.59      0.47      0.45      4755


_____
__
```

In [ ]:
```
# 8. verbs & lemmatized
print("Only verbs & lemmatized")
run_vader_on_airline_tweets(to_lemmatize=True, pos={'VERB'})
print("_____
```

```
Only verbs & lemmatized
              precision    recall  f1-score   support

    negative       0.74      0.30      0.42      1750
     neutral       0.38      0.78      0.51      1515
    positive       0.57      0.35      0.43      1490

    accuracy                           0.47      4755
   macro avg       0.56      0.48      0.46      4755
weighted avg       0.57      0.47      0.45      4755


_____
__
```

Part II: scikit-learn assignments

[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?

b.

- Just like in question 4, the difference between td-idf and bag of words representation doesnt seem to differ significantly enough to infer either being superior to the other. This goes for any of the min_df setting, although the small differences that do occur do differ between the settings.
- between a min_df of 2 and 5 there is very little noticeable difference. however, between 5 and 10 there is a larger difference. This is likely because the threshold of 10 is much harsher than that of 5, and will thus exclude much more words from consideration. This seems to, in this case, affect negative words the most in terms of precision and neutral the most in terms of recall.

In [ ]:
```
# Your code here
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
import nltk
from sklearn.feature_extraction.text import TfidfTransformer
import numpy as np
```

In [ ]:
```
airline_vec = CountVectorizer(min_df=2, # If a token appears fewer times than this,
                              tokenizer=nltk.word_tokenize, # we use the nltk tokeniz
                              stop_words=stopwords.words('english')) # stopwords are

airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

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 development
    )

clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

#generate a classification report
print(classification_report(y_test, y_pred, target_names=airline_tweets_train.targe
```

```
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern
' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be incon
sistent with your preprocessing. Tokenizing the stop words generated tokens ["'d", "
'll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't", 'need', 'sha', 'wo', 'wou
ld'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

    negative       0.82      0.92      0.86       337
     neutral       0.85      0.70      0.77       323
    positive       0.83      0.87      0.85       291

    accuracy                           0.83       951
   macro avg       0.83      0.83      0.83       951
weighted avg       0.83      0.83      0.83       951
```

In [ ]:
```
airline_vec = CountVectorizer(min_df=2, # If a token appears fewer times than this,
                              tokenizer=nltk.word_tokenize, # we use the nltk tokeniz
                              stop_words=stopwords.words('english')) # stopwords are

airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # 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 development
    )

clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

#generate a classification report
print(classification_report(y_test, y_pred, target_names=airline_tweets_train.targe
```

```
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern
' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be incon
sistent with your preprocessing. Tokenizing the stop words generated tokens ["'d", "
'll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't", 'need', 'sha', 'wo', 'wou
ld'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

    negative       0.85      0.90      0.87       353
     neutral       0.84      0.74      0.79       308
    positive       0.83      0.88      0.85       290

    accuracy                           0.84       951
   macro avg       0.84      0.84      0.84       951
weighted avg       0.84      0.84      0.84       951
```

In [ ]:
```
airline_vec = CountVectorizer(min_df=5, # If a token appears fewer times than this,
                              tokenizer=nltk.word_tokenize, # we use the nltk tokeniz
                              stop_words=stopwords.words('english')) # stopwords are

airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

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 development
    )

clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

#generate a classification report
print(classification_report(y_test, y_pred, target_names=airline_tweets_train.targe
```

```
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern
' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be incon
sistent with your preprocessing. Tokenizing the stop words generated tokens ["'d", "
'll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't", 'need', 'sha', 'wo', 'wou
ld'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

    negative       0.82      0.90      0.86       346
     neutral       0.87      0.73      0.80       313
    positive       0.85      0.89      0.87       292

    accuracy                           0.84       951
   macro avg       0.85      0.84      0.84       951
weighted avg       0.84      0.84      0.84       951
```

In [ ]:
```
airline_vec = CountVectorizer(min_df=5, # If a token appears fewer times than this,
                              tokenizer=nltk.word_tokenize, # we use the nltk tokeniz
                              stop_words=stopwords.words('english')) # stopwords are

airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # 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 development
    )

clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

#generate a classification report
print(classification_report(y_test, y_pred, target_names=airline_tweets_train.targe
```

```
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern
' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be incon
sistent with your preprocessing. Tokenizing the stop words generated tokens ["'d", "
'll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't", 'need', 'sha', 'wo', 'wou
ld'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

    negative       0.84      0.90      0.87       365
     neutral       0.79      0.74      0.76       288
    positive       0.83      0.81      0.82       298

    accuracy                           0.83       951
   macro avg       0.82      0.82      0.82       951
weighted avg       0.82      0.83      0.82       951
```

In [ ]:
```
airline_vec = CountVectorizer(min_df=10, # If a token appears fewer times than this
                              tokenizer=nltk.word_tokenize, # we use the nltk tokeniz
                              stop_words=stopwords.words('english')) # stopwords are

airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

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 development
    )

clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

#generate a classification report
print(classification_report(y_test, y_pred, target_names=airline_tweets_train.targe
```

```
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern
' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be incon
sistent with your preprocessing. Tokenizing the stop words generated tokens ["'d", "
'll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't", 'need', 'sha', 'wo', 'wou
ld'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

    negative       0.79      0.90      0.84       326
     neutral       0.81      0.70      0.75       313
    positive       0.83      0.82      0.83       312

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

In [ ]:
```
airline_vec = CountVectorizer(min_df=10, # If a token appears fewer times than this
                              tokenizer=nltk.word_tokenize, # we use the nltk tokeniz
                              stop_words=stopwords.words('english')) # stopwords are

airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # 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 development
    )

clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

#generate a classification report
print(classification_report(y_test, y_pred, target_names=airline_tweets_train.targe
```

```
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern
' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be incon
sistent with your preprocessing. Tokenizing the stop words generated tokens ["'d", "
'll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't", 'need', 'sha', 'wo', 'wou
ld'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

    negative       0.83      0.93      0.88       372
     neutral       0.82      0.71      0.76       295
    positive       0.84      0.82      0.83       284

    accuracy                           0.83       951
   macro avg       0.83      0.82      0.82       951
weighted avg       0.83      0.83      0.83       951
```

[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?

In [ ]:

b. Negative:

- We expected words like cancelled, delayed, late, worst, waiting, terrible, delay, lost and exclamations such as #, ! and ?. These are words often used when someone is trying to portray they are mad, or negative things that can happen in the context of flying.
- Words like thanks and things such as number or punctuation marks strike us as weird in a list of negative words. These things dont have an inherently negative connotation with them, thus you would not expect them to be such important features when trying to extract negativity.
- Words that are just very common in this domain should be removed, as these will not have a very high impact on the final label, even though they are always common. Think here of words like flight, service, airline, back, fly, etc.

Neutral:

- words like please, help, tomorrow, change, flying, reservation, ticket, in short, any word that either points to just general aviation related things or any word that is part of a question. These are both not inherently positive or negative. It seems that most of the neutral tweets are questions or tracking-accounts in this dataset.
- for example, cancelled was not expected in this list, as this is usually a negative word.
- As opposed to with the negative list, we would remove words that have a very high sentimental score. Words such as happy, or cancelled, etc. are usually not used in neutral sentences (although they could be) and should thus mostly not be used to try and classify neutral tweets.

Positive:

- Expected words are thanks, great, love, best, awesome, good, amazing, etc. These words are very positive and, unless used sarcastically, are always used in very positive tweets.
- We did not expect words like please, because they are often either used in a question (neutral) or in a sarcastic way to say they are not satisfied (negative). We did not expect this to be in the positive features for that reason.
- As in the negative features, we would remove neutral words such as flying, customer, etc. The explanation for this is the same -> these do not necessarily point to either positive or negative sentiment.

In [ ]:
```
airline_vec = CountVectorizer(min_df=2, # If a token appears fewer times than this,
                              tokenizer=nltk.word_tokenize, # we use the nltk tokeniz
                              stop_words=stopwords.words('english')) # stopwords are

airline_counts = airline_vec.fit_transform(airline_tweets_train.data)

tfidf_transformer = TfidfTransformer()
airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

docs_train, docs_test, y_train, y_test = train_test_split(
    airline_counts, # 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 development
    )

clf = MultinomialNB().fit(docs_train, y_train)
y_pred = clf.predict(docs_test)

#generate a classification report
print(classification_report(y_test, y_pred, target_names=airline_tweets_train.targe
```

```
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern
' will not be used since 'tokenizer' is not None'
  warnings.warn(
c:\Users\Kelvin Brachthuizen\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be incon
sistent with your preprocessing. Tokenizing the stop words generated tokens ["'d", "
'll", "'re", "'s", "'ve", 'could', 'might', 'must', "n't", 'need', 'sha', 'wo', 'wou
ld'] not in stop_words.
  warnings.warn(
              precision    recall  f1-score   support

    negative       0.88      0.90      0.89       361
     neutral       0.84      0.75      0.79       292
    positive       0.83      0.89      0.86       298

    accuracy                           0.85       951
   macro avg       0.85      0.85      0.85       951
weighted avg       0.85      0.85      0.85       951
```

In [ ]:
```
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=T
    topn_class2 = sorted(zip(classifier.feature_count_[1], feature_names),reverse=T
    topn_class3 = sorted(zip(classifier.feature_count_[2], feature_names),reverse=T
    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 1481.0 @
0 1373.0 united
0 1219.0 .
0 425.0 ``
0 396.0 flight
0 385.0 ?
0 367.0 !
0 314.0 #
0 214.0 n't
0 166.0 ''
0 118.0 service
0 115.0 's
0 104.0 virginamerica
0 102.0 :
0 96.0 get
0 96.0 cancelled
0 92.0 delayed
0 90.0 customer
0 86.0 bag
0 81.0 time
0 79.0 ;
0 78.0 plane
0 75.0 -
0 74.0 ...
0 74.0 'm
0 72.0 &
0 70.0 hours
0 69.0 gate
0 64.0 http
0 63.0 still
0 62.0 hour
0 61.0 amp
0 59.0 late
0 59.0 airline
0 57.0 help
0 56.0 2
0 55.0 would
0 53.0 ca
0 52.0 like
0 49.0 worst
0 49.0 waiting
0 48.0 flights
0 48.0 delay
0 48.0 $
0 47.0 never
0 46.0 flightled
0 46.0 back
0 45.0 one
0 44.0 've
0 43.0 fly
0 42.0 us
0 41.0 (
0 39.0 seat
0 37.0 ever
0 37.0 3
```

```
0 37.0 )
0 35.0 wait
0 35.0 really
0 35.0 due
0 34.0 thanks
0 34.0 luggage
0 34.0 lost
0 34.0 going
0 34.0 day
0 33.0 trying
0 33.0 hold
0 33.0 another
0 31.0 u
0 31.0 ticket
0 31.0 last
0 31.0 guys
0 31.0 check
0 30.0 seats
0 30.0 people
0 30.0 4
0 29.0 terrible
0 29.0 even
0 29.0 days
0 29.0 could
0 28.0 staff
----------------------------------------
Important words in neutral documents
1 1405.0 @
1 549.0 .
1 519.0 ?
1 311.0 jetblue
1 293.0 :
1 278.0 united
1 267.0 southwestair
1 243.0 #
1 236.0 ``
1 229.0 flight
1 186.0 americanair
1 185.0 http
1 168.0 !
1 159.0 usairways
1 148.0 's
1 85.0 get
1 74.0 -
1 67.0 virginamerica
1 65.0 please
1 65.0 flights
1 59.0 help
1 59.0 ''
1 57.0 n't
1 54.0 )
1 53.0 need
1 53.0 ...
1 50.0 ;
1 49.0 (
1 46.0 dm
```

```
1 43.0 &
1 42.0 us
1 42.0 fleet
1 42.0 fleek
1 41.0 tomorrow
1 40.0 “
1 39.0 ”
1 38.0 would
1 38.0 know
1 35.0 thanks
1 35.0 change
1 33.0 flying
1 33.0 amp
1 32.0 hi
1 31.0 one
1 30.0 way
1 30.0 fly
1 29.0 could
1 29.0 cancelled
1 28.0 'm
1 27.0 travel
1 27.0 number
1 26.0 today
1 26.0 like
1 25.0 go
1 25.0 check
1 24.0 time
1 24.0 tickets
1 23.0 new
1 23.0 airport
1 22.0 use
1 22.0 sent
1 22.0 see
1 22.0 rt
1 22.0 destinationdragons
1 21.0 want
1 20.0 ticket
1 20.0 make
1 20.0 follow
1 20.0 chance
1 20.0 2
1 19.0 reservation
1 19.0 next
1 19.0 bag
1 19.0 add
1 18.0 vegas
1 18.0 question
1 18.0 guys
1 18.0 gate
1 17.0 still
1 17.0 seat
----------------------------------------
Important words in positive documents
2 1339.0 @
2 1067.0 !
2 740.0 .
```

```
2 321.0 #
2 312.0 southwestair
2 282.0 thanks
2 282.0 jetblue
2 261.0 united
2 258.0 thank
2 250.0 ``
2 173.0 flight
2 173.0 americanair
2 167.0 :
2 134.0 usairways
2 129.0 great
2 92.0 service
2 80.0 )
2 79.0 virginamerica
2 74.0 love
2 74.0 http
2 69.0 customer
2 68.0 guys
2 68.0 ;
2 64.0 best
2 64.0 's
2 60.0 much
2 54.0 awesome
2 51.0 -
2 49.0 got
2 49.0 good
2 49.0 &
2 48.0 amazing
2 46.0 airline
2 43.0 amp
2 40.0 today
2 40.0 help
2 38.0 us
2 38.0 time
2 38.0 n't
2 37.0 fly
2 35.0 get
2 35.0 flying
2 32.0 crew
2 30.0 gate
2 30.0 ''
2 29.0 made
2 29.0 back
2 29.0 appreciate
2 29.0 ...
2 28.0 response
2 27.0 see
2 27.0 'm
2 26.0 work
2 26.0 home
2 26.0 ever
2 25.0 plane
2 25.0 day
2 25.0 ?
2 25.0 're
```

```
2 23.0 would
2 23.0 nice
2 23.0 new
2 23.0 like
2 23.0 always
2 22.0 well
2 22.0 tonight
2 22.0 staff
2 21.0 follow
2 20.0 u
2 20.0 team
2 20.0 please
2 20.0 make
2 19.0 yes
2 19.0 wait
2 19.0 southwest
2 19.0 really
2 19.0 know
2 19.0 job
2 19.0 happy
2 19.0 flights
```

[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.

[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.

End of this notebook

In [ ]:
```
:)
```