CSML1010 Day 2 coding exercise

Second attempt, Dec 4 2019

Pete Gray

I wasn't able to even crack this one on my old computers.

Emotion and Sentiment Analysis

Sentiment analysis is perhaps one of the most popular applications of NLP, with a vast number of tutorials, courses, and applications that focus on analyzing sentiments of diverse datasets ranging from corporate surveys to movie reviews. The key aspect of sentiment analysis is to analyze a body of text for understanding the opinion expressed by it. Typically, we quantify this sentiment with a positive or negative value, called polarity. The overall sentiment is often inferred as positive, neutral or negative from the sign of the polarity score.

Usually, sentiment analysis works best on text that has a subjective context than on text with only an objective context. Objective text usually depicts some normal statements or facts without expressing any emotion, feelings, or mood. Subjective text contains text that is usually expressed by a human having typical moods, emotions, and feelings. Sentiment analysis is widely used, especially as a part of social media analysis for any domain, be it a business, a recent movie, or a product launch, to understand its reception by the people and what they think of it based on their opinions or, you guessed it, sentiment!

Typically, sentiment analysis for text data can be computed on several levels, including on an individual sentence level, paragraph level, or the entire document as a whole. Often, sentiment is computed on the document as a whole or some aggregations are done after computing the sentiment for individual sentences. There are two major approaches to sentiment analysis.

- Supervised machine learning or deep learning approaches
- Unsupervised lexicon-based approaches

For the first approach we typically need pre-labeled data. Hence, we will be focusing on the second approach. For a comprehensive coverage of sentiment analysis, refer to Chapter 7: Analyzing Movie Reviews Sentiment, Practical Machine Learning with Python, Springer\Apress, 2018. In this scenario, we do not have the convenience of a well-labeled training dataset. Hence, we will need to use unsupervised techniques for predicting the sentiment by using knowledgebases, ontologies, databases, and lexicons that have detailed information, specially curated and prepared just for sentiment analysis. A lexicon is a dictionary, vocabulary, or a book of words. In our case, lexicons are special dictionaries or vocabularies that have been created for analyzing sentiments. Most of these lexicons have a list of positive and negative polar words with some score associated with them, and using various techniques like the position of words, surrounding words, context, parts of speech, phrases, and so on, scores are assigned to the text documents for which we want to compute the sentiment. After aggregating these scores, we get the final sentiment.

Various popular lexicons are used for sentiment analysis, including the following.

AFINN lexicon Bing Liu's lexicon MPQA subjectivity lexicon SentiWordNet VADER lexicon TextBlob lexicon This is not an exhaustive list of lexicons that can be leveraged for sentiment analysis, and there are several other lexicons which can be easily obtained from the Internet. Feel free to check out each of these links and explore them. We will be covering two techniques in this section.

Some Pre-Processing

Import necessary depencencies

In [1]:
```
import pandas as pd
import numpy as np
import text_normalizer as tn
import model_evaluation_utils as meu

np.set_printoptions(precision=2, linewidth=80)
```

Load and normalize data

1. Cleaning Text - strip HTML
2. Removing accented characters
3. Expanding Contractions
4. Removing Special Characters
5. Lemmatizing text¶
6. Removing Stopwords

In [5]:
```
dataset = pd.read_csv(r'movie_reviews_cleaned.csv')

reviews = np.array(dataset['review'])
sentiments = np.array(dataset['sentiment'])

# extract data for model evaluation
train_reviews = reviews[:10000]
train_sentiments = sentiments[:10000]

test_reviews = reviews[10000:15000]
test_sentiments = sentiments[10000:15000]
sample_review_ids = [2626, 3533, 3010]
```

In [ ]:
```
# SKIP FOR THE STUDENTS BECAUSE INSTRUCTOR HAS PRE_NORMALIZED AND SAVED THE FILE
# normalize dataset (time consuming using spacey pipeline)
"""
norm_test_reviews = tn.normalize_corpus(test_reviews)
norm_train_reviews = tn.normalize_corpus(train_reviews)
#output back to a csv file again
import csv
with open(r'movie_reviews_cleaned.csv', mode='w') as cleaned_file:
    csv_writer = csv.writer(cleaned_file, delimiter=',', quotechar='"', quoting=c
    csv_writer.writerow(['review', 'sentiment'])
    for  text, sent in zip(norm_test_reviews, test_sentiments):
        csv_writer.writerow([text, sent])
    for  text, sent in zip(norm_train_reviews, train_sentiments):
        csv_writer.writerow([text, sent])
"""
```

==

Part A. Unsupervised (Lexicon) Sentiment Analysis

==

1. Sentiment Analysis with AFINN

The AFINN lexicon is perhaps one of the simplest and most popular lexicons that can be used extensively for sentiment analysis. Developed and curated by Finn Arup Nielsen, you can find more details on this lexicon in the paper, “A new ANEW: evaluation of a word list for sentiment analysis in microblogs”, proceedings of the ESWC 2011 Workshop. The current version of the lexicon is AFINN-en-165. txt and it contains over 3,300+ words with a polarity score associated with each word. You can find this lexicon at the author’s official GitHub repository along with previous versions of it, including AFINN-111. The author has also created a nice wrapper library on top of this in Python called afinn, which we will be using for our analysis.

In [3]:
```
from afinn import Afinn

afn = Afinn(emoticons=True)

# NOTE:  to use afinn score, call the function afn.score("text you want the sent
# the lexicon will be used to compute summary of sentiment for the given text
```

Predict sentiment for sample reviews

We can get a good idea of general sentiment for different sample.

In [6]:
```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[sa
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    print('Predicted Sentiment polarity:', afn.score(review))
    print('-'*60)
```

```
REVIEW: film producer hop cameron diaz name help sell picture unfortunately not
hing save already capture screen despite beautifully shoot european location so
lid production design element film fail mostly due awkward unbelievable romance
brewster eccleston unplesasant filmgoing experience
Actual Sentiment: negative
Predicted Sentiment polarity: 5.0
------------------------------------------------------------
REVIEW: totally surprised comment forum many review think tony scott make good
movie yes highly stylize flashy top entertaining glad least ebert roeper agree
movie may not anyone like top dark humor cool action dialog see previously see
scott man fire crimson tide enemy state good movie like one like roller coaster
ride great soundtrack selection visual style time movie seem pg13 nice see some
one not afraid show nudity gory violence explicit dialog not hurt keira super h
ot even show nipple one either
Actual Sentiment: positive
Predicted Sentiment polarity: 24.0
------------------------------------------------------------
REVIEW: bad horror film ever funniest film ever roll one get see film cheap unb
eliaveble see really p watch carrot
Actual Sentiment: positive
Predicted Sentiment polarity: -3.0
------------------------------------------------------------
```

Predict sentiment for test dataset

In [7]:
```
sentiment_polarity = [afn.score(review) for review in test_reviews]
predicted_sentiments = ['positive' if score >= 1.0 else 'negative' for score in s
```

Evaluate model performance

In [8]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_labe
                                      classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.713
Precision: 0.7333
Recall: 0.713
F1 Score: 0.7074

Model Classification report:
------------------------------
              precision    recall  f1-score   support

    positive       0.66      0.86      0.75      2470
    negative       0.80      0.57      0.67      2530

    accuracy                           0.71      5000
   macro avg       0.73      0.71      0.71      5000
weighted avg       0.73      0.71      0.71      5000


Prediction Confusion Matrix:
------------------------------

C:\Users\Dell\CSML1010\nlp-lifecycle-project-fall2019\exercises\day2\model_eval
uation_utils.py:62: FutureWarning: the 'labels' keyword is deprecated, use 'cod
es' instead
  labels=level_labels),
C:\Users\Dell\CSML1010\nlp-lifecycle-project-fall2019\exercises\day2\model_eval
uation_utils.py:64: FutureWarning: the 'labels' keyword is deprecated, use 'cod
es' instead
  labels=level_labels))

                 Predicted:
                   positive negative
Actual: positive       2113      357
        negative       1078     1452
```

2. Sentiment Analysis with SentiWordNet

SentiWordNet is a lexical resource for opinion mining. SentiWordNet assigns to each synset of WordNet three sentiment scores: positivity, negativity, objectivity. SentiWordNet is described in details in the papers:

In [9]:
```
from nltk.corpus import sentiwordnet as swn
import nltk
nltk.download('sentiwordnet')

awesome = list(swn.senti_synsets('awesome', 'a'))[0]
print('Positive Polarity Score:', awesome.pos_score())
print('Negative Polarity Score:', awesome.neg_score())
print('Objective Score:', awesome.obj_score())
```

```
[nltk_data] Downloading package sentiwordnet to
[nltk_data]     C:\Users\Dell\AppData\Roaming\nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!

Positive Polarity Score: 0.875
Negative Polarity Score: 0.125
Objective Score: 0.0
```

Build model

For each word in the review, add up the sentiment score of words that are NN, VB, JJ, RB if it's in the lexicon dictionary.

In [10]:
```
def analyze_sentiment_sentiwordnet_lexicon(review,
                                           verbose=False):

    # tokenize and POS tag text tokens
    tagged_text = [(token.text, token.tag_) for token in tn.nlp(review)]
    pos_score = neg_score = token_count = obj_score = 0
    # get wordnet synsets based on POS tags
    # get sentiment scores if synsets are found
    for word, tag in tagged_text:
        ss_set = None
        if 'NN' in tag and list(swn.senti_synsets(word, 'n')):
            ss_set = list(swn.senti_synsets(word, 'n'))[0]
        elif 'VB' in tag and list(swn.senti_synsets(word, 'v')):
            ss_set = list(swn.senti_synsets(word, 'v'))[0]
        elif 'JJ' in tag and list(swn.senti_synsets(word, 'a')):
            ss_set = list(swn.senti_synsets(word, 'a'))[0]
        elif 'RB' in tag and list(swn.senti_synsets(word, 'r')):
            ss_set = list(swn.senti_synsets(word, 'r'))[0]
        # if senti-synset is found
        if ss_set:
            # add scores for all found synsets
            pos_score += ss_set.pos_score()
            neg_score += ss_set.neg_score()
            obj_score += ss_set.obj_score()
            token_count += 1

    # aggregate final scores
    final_score = pos_score - neg_score
    norm_final_score = round(float(final_score) / token_count, 2)
    final_sentiment = 'positive' if norm_final_score >= 0 else 'negative'
    if verbose:
        norm_obj_score = round(float(obj_score) / token_count, 2)
        norm_pos_score = round(float(pos_score) / token_count, 2)
        norm_neg_score = round(float(neg_score) / token_count, 2)
        # to display results in a nice table
        sentiment_frame = pd.DataFrame([[final_sentiment, norm_obj_score, norm_p
                                         norm_neg_score, norm_final_score]],
                                       columns=pd.MultiIndex(levels=[['SENTIMENT
                                                             ['Predicted Sentime
                                                              'Positive', 'Negat
                                                             labels=[[0,0,0,0,0]
        print(sentiment_frame)

    return final_sentiment
```

Predict sentiment for sample reviews

In [11]:
```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[sa
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    pred = analyze_sentiment_sentiwordnet_lexicon(review, verbose=True)
    print('-'*60)
```

```
REVIEW: film producer hop cameron diaz name help sell picture unfortunately not
hing save already capture screen despite beautifully shoot european location so
lid production design element film fail mostly due awkward unbelievable romance
brewster eccleston unplesasant filmgoing experience
Actual Sentiment: negative

C:\Users\Dell\Anaconda3\lib\site-packages\ipykernel_launcher.py:41: FutureWarni
ng: the 'labels' keyword is deprecated, use 'codes' instead

     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.84     0.09     0.07    0.03
------------------------------------------------------------
REVIEW: totally surprised comment forum many review think tony scott make good
movie yes highly stylize flashy top entertaining glad least ebert roeper agree
movie may not anyone like top dark humor cool action dialog see previously see
scott man fire crimson tide enemy state good movie like one like roller coaster
ride great soundtrack selection visual style time movie seem pg13 nice see some
one not afraid show nudity gory violence explicit dialog not hurt keira super h
ot even show nipple one either
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.84     0.08     0.07    0.01
------------------------------------------------------------
REVIEW: bad horror film ever funniest film ever roll one get see film cheap unb
eliaveble see really p watch carrot
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.86     0.07     0.07     0.0
------------------------------------------------------------
```

Predict sentiment for test dataset

In [13]:
```
predicted_sentiments = [analyze_sentiment_sentiwordnet_lexicon(review, verbose=F
```

Evaluate model performance

In [14]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_labe
                                  classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.6736
Precision: 0.6792
Recall: 0.6736
F1 Score: 0.6716

Model Classification report:
------------------------------
              precision    recall  f1-score   support

    positive       0.65      0.75      0.70      2470
    negative       0.71      0.59      0.65      2530

    accuracy                           0.67      5000
   macro avg       0.68      0.67      0.67      5000
weighted avg       0.68      0.67      0.67      5000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       1863      607
        negative       1025     1505
```

3. Sentiment Analysis with VADER

In [15]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

Build model

In [16]:
```
def analyze_sentiment_vader_lexicon(review,
                                    threshold=0.1,
                                    verbose=False):
    # pre-process text
    review = tn.strip_html_tags(review)
    review = tn.remove_accented_chars(review)
    review = tn.expand_contractions(review)

    # analyze the sentiment for review
    analyzer = SentimentIntensityAnalyzer()
    scores = analyzer.polarity_scores(review)
    # get aggregate scores and final sentiment
    agg_score = scores['compound']
    final_sentiment = 'positive' if agg_score >= threshold\
                                   else 'negative'
    if verbose:
        # display detailed sentiment statistics
        positive = str(round(scores['pos'], 2)*100)+'%'
        final = round(agg_score, 2)
        negative = str(round(scores['neg'], 2)*100)+'%'
        neutral = str(round(scores['neu'], 2)*100)+'%'
        sentiment_frame = pd.DataFrame([[final_sentiment, final, positive,
                                        negative, neutral]],
                                        columns=pd.MultiIndex(levels=[['SENTIMEN
                                                                      ['Predicte
                                                                       'Positive
                                                              labels=[[0,0,0,0,0
        print(sentiment_frame)

    return final_sentiment
```

Predict sentiment for sample reviews

In [18]:
```
nltk.download('vader_lexicon')
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[sa
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    pred = analyze_sentiment_vader_lexicon(review, threshold=0.4, verbose=True)
    print('-'*60)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Dell\AppData\Roaming\nltk_data...

REVIEW: film producer hop cameron diaz name help sell picture unfortunately not
hing save already capture screen despite beautifully shoot european location so
lid production design element film fail mostly due awkward unbelievable romance
brewster eccleston unplesasant filmgoing experience
Actual Sentiment: negative

C:\Users\Dell\Anaconda3\lib\site-packages\ipykernel_launcher.py:27: FutureWarni
ng: the 'labels' keyword is deprecated, use 'codes' instead

     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative Neutral
0            negative          -0.34    23.0%    26.0%   51.0%
------------------------------------------------------------
REVIEW: totally surprised comment forum many review think tony scott make good
movie yes highly stylize flashy top entertaining glad least ebert roeper agree
movie may not anyone like top dark humor cool action dialog see previously see
scott man fire crimson tide enemy state good movie like one like roller coaster
ride great soundtrack selection visual style time movie seem pg13 nice see some
one not afraid show nudity gory violence explicit dialog not hurt keira super h
ot even show nipple one either
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative Neutral
0            positive           0.94    32.0%    17.0%   52.0%
------------------------------------------------------------
REVIEW: bad horror film ever funniest film ever roll one get see film cheap unb
eliaveble see really p watch carrot
Actual Sentiment: positive
     SENTIMENT STATS:                                       \
  Predicted Sentiment Polarity Score             Positive
0            negative          -0.56  14.000000000000002%


              Negative             Neutral
0  28.000000000000004%  57.99999999999999%
------------------------------------------------------------
```

Predict sentiment for test dataset

In [19]:
```
predicted_sentiments = [analyze_sentiment_vader_lexicon(review, threshold=0.4, v
```

Evaluate model performance

In [20]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_lab
                                  classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.7038
Precision: 0.7152
Recall: 0.7038
F1 Score: 0.7005

Model Classification report:
------------------------------
              precision    recall  f1-score   support

    positive       0.66      0.81      0.73      2470
    negative       0.77      0.60      0.67      2530

    accuracy                           0.70      5000
   macro avg       0.71      0.71      0.70      5000
weighted avg       0.72      0.70      0.70      5000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       2008      462
        negative       1019     1511
```

#
#
End of part 1 - Unsupervised
#
