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 [2]:
```
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[:35000]
train_sentiments = sentiments[:35000]

test_reviews = reviews[35000:]
test_sentiments = sentiments[35000:]
sample_review_ids = [7626, 3533, 13010]
```

In [3]:
```
# SKIP FOR THE STUDENTS BECAUSE INSTRUCTOR HAS PRE_NORMALIZED AND SAVED THE FI
LE
# 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='"', quotin
g=csv.QUOTE_MINIMAL)
    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])
"""
```

Out[3]:
```
'\nnorm_test_reviews = tn.normalize_corpus(test_reviews)\nnorm_train_reviews
= tn.normalize_corpus(train_reviews)\n#output back to a csv file again\nimpor
t csv\nwith open(r\'movie_reviews_cleaned.csv\', mode=\'w\') as cleaned_fil
e:\n    csv_writer = csv.writer(cleaned_file, delimiter=\',\', quotechar=
\'"\', quoting=csv.QUOTE_MINIMAL)\n    csv_writer.writerow([\'review\', \'sen
timent\'])\n    for  text, sent in zip(norm_test_reviews, test_sentiments):\n
csv_writer.writerow([text, sent])\n    for  text, sent in zip(norm_train_revi
ews, train_sentiments):\n        csv_writer.writerow([text, sent])\n'
```

==

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 [4]:
```
from afinn import Afinn

afn = Afinn(emoticons=True)

# NOTE:  to use afinn score, call the function afn.score("text you want the se
ntiment for")
# 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 [5]:
```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[
sample_review_ids]):
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    print('Predicted Sentiment polarity:', afn.score(review))
    print('-'*60)
```

```
REVIEW: word fail whenever want describe feeling movie sequel flaw sure start
subspecie not execute well enough special effect glorify movie herd movie mas
s consumer care quantity quality cheap fun depth crap like blade not even des
erve capital letter underworlddracula 2000dracula 3000 good movie munch popco
rn drink couple coke make subspecie superior effort anyone claim vampire fana
tic hand obvious vampire romanian story set transylvania scene film location
convince atmosphere not base action pack chase expensive orchestral music rad
u source atmosphere vampire look like behave add breathtakingly gloomy castle
dark passageway situate romania include typical vampiric element movement sha
dow wall vampire take flight work art short like fascinated vampire feel appe
arance well setting sinister dark no good place look subspecie movie vampire
journal brilliant spin former
Actual Sentiment: positive
Predicted Sentiment polarity: 20.0
------------------------------------------------------------
REVIEW: good family movie laugh wish not much school stuff like bully fill mo
vie also seem little easy save piece land build mean flow easily make aware w
ildlife cute way introduce piece land fast runner little slow little hokey re
mind go back school oh dvd chock full goody not miss 7 10 movie 10 10 dvd ext
ra well worth watch well worth time see
Actual Sentiment: positive
Predicted Sentiment polarity: 12.0
------------------------------------------------------------
REVIEW: opinion movie not good hardly find good thing say still would like ex
plain conclude another bad movie decide watch costas mandylor star main reaso
n watch till end like action movie understand movie build action rather story
know not go detail come credibility story event even not explain scene lack s
ense reality look ridiculous beginning movie look quite promising tough good
look specialist not tough smart funny partner must job turn bit different exp
ect story take place cruise ship disaster happen ship turn leave alive strugg
le survive escape shark professional killer rise water furthermore movie quit
e violent main weapon beside disaster already take passenger gun successfully
use many case personally miss good man man woman woman prefer fight family fu
n not think think movie shoot hurry without real vision try say make usual ac
tion movie trick bit something call love without real meaning result bad movi
e
Actual Sentiment: negative
Predicted Sentiment polarity: 2.0
------------------------------------------------------------
```

Predict sentiment for test dataset

In [6]:
```
sentiment_polarity = [afn.score(review) for review in test_reviews]
predicted_sentiments = ['positive' if score >= 1.0 else 'negative' for score i
n sentiment_polarity]
```

Evaluate model performance

In [7]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_l
abels=predicted_sentiments,
                                  classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.7054
Precision: 0.7212
Recall: 0.7054
F1 Score: 0.6993

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

    positive       0.66      0.84      0.74      7587
    negative       0.78      0.56      0.65      7413

   micro avg       0.71      0.71      0.71     15000
   macro avg       0.72      0.70      0.70     15000
weighted avg       0.72      0.71      0.70     15000


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

C:\Users\finan\Downloads\ML1010_InClass-master\ML1010_InClass-master\Day1\3_s
entiment\model_evaluation_utils.py:62: FutureWarning: the 'labels' keyword is
deprecated, use 'codes' instead
  labels=level_labels),
C:\Users\finan\Downloads\ML1010_InClass-master\ML1010_InClass-master\Day1\3_s
entiment\model_evaluation_utils.py:64: FutureWarning: the 'labels' keyword is
deprecated, use 'codes' instead
  labels=level_labels))

                 Predicted:
                   positive negative
Actual: positive       6405     1182
        negative       3237     4176
```

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 [8]:
```
from nltk.corpus import sentiwordnet as swn
import nltk
nltk.download('sentiwordnet')
nltk.download('wordnet')

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\finan\AppData\Roaming\nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\finan\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.

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 [9]:
```
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
_pos_score,
                                         norm_neg_score, norm_final_score]],
                                       columns=pd.MultiIndex(levels=[['SENTIME
NT STATS:'],
                                                              ['Predicted Senti
ment', 'Objectivity',
                                                               'Positive', 'Neg
ative', 'Overall']],
                                                              labels=[[0,0,0,0,
0],[0,1,2,3,4]]))
        print(sentiment_frame)

    return final_sentiment
```

Predict sentiment for sample reviews

In [10]:
```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[
sample_review_ids]):
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    pred = analyze_sentiment_sentiwordnet_lexicon(review, verbose=True)
    print('-'*60)
```

```
REVIEW: word fail whenever want describe feeling movie sequel flaw sure start
subspecie not execute well enough special effect glorify movie herd movie mas
s consumer care quantity quality cheap fun depth crap like blade not even des
erve capital letter underworlddracula 2000dracula 3000 good movie munch popco
rn drink couple coke make subspecie superior effort anyone claim vampire fana
tic hand obvious vampire romanian story set transylvania scene film location
convince atmosphere not base action pack chase expensive orchestral music rad
u source atmosphere vampire look like behave add breathtakingly gloomy castle
dark passageway situate romania include typical vampiric element movement sha
dow wall vampire take flight work art short like fascinated vampire feel appe
arance well setting sinister dark no good place look subspecie movie vampire
journal brilliant spin former
Actual Sentiment: positive

c:\users\finan\appdata\local\programs\python\python37\lib\site-packages\ipyke
rnel_launcher.py:41: FutureWarning: the 'labels' keyword is deprecated, use
'codes' instead

     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.85     0.09     0.06    0.02
------------------------------------------------------------
REVIEW: good family movie laugh wish not much school stuff like bully fill mo
vie also seem little easy save piece land build mean flow easily make aware w
ildlife cute way introduce piece land fast runner little slow little hokey re
mind go back school oh dvd chock full goody not miss 7 10 movie 10 10 dvd ext
ra well worth watch well worth time see
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.85     0.08     0.08     0.0
------------------------------------------------------------
REVIEW: opinion movie not good hardly find good thing say still would like ex
plain conclude another bad movie decide watch costas mandylor star main reaso
n watch till end like action movie understand movie build action rather story
know not go detail come credibility story event even not explain scene lack s
ense reality look ridiculous beginning movie look quite promising tough good
look specialist not tough smart funny partner must job turn bit different exp
ect story take place cruise ship disaster happen ship turn leave alive strugg
le survive escape shark professional killer rise water furthermore movie quit
e violent main weapon beside disaster already take passenger gun successfully
use many case personally miss good man man woman woman prefer fight family fu
n not think think movie shoot hurry without real vision try say make usual ac
tion movie trick bit something call love without real meaning result bad movi
e
Actual Sentiment: negative
     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.82     0.09     0.09     0.0
------------------------------------------------------------
```

Predict sentiment for test dataset

In [11]:
```
predicted_sentiments = [analyze_sentiment_sentiwordnet_lexicon(review, verbose
=False) for review in test_reviews]
```

Evaluate model performance

In [12]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_l
abels=predicted_sentiments,
                                  classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.6812
Precision: 0.6846
Recall: 0.6812
F1 Score: 0.6792

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

    positive       0.66      0.76      0.71      7587
    negative       0.71      0.60      0.65      7413

   micro avg       0.68      0.68      0.68     15000
   macro avg       0.68      0.68      0.68     15000
weighted avg       0.68      0.68      0.68     15000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       5741     1846
        negative       2936     4477
```

3. Sentiment Analysis with VADER

In [13]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

Build model

In [14]:
```
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=[['SENTIM
ENT STATS:'],
                                                                      ['Predic
ted Sentiment', 'Polarity Score',
                                                                       'Positi
ve', 'Negative', 'Neutral']],
                                                              labels=[[0,0,0,0
,0],[0,1,2,3,4]]))
        print(sentiment_frame)

    return final_sentiment
```

Predict sentiment for sample reviews

In [16]:
```
nltk.download('vader_lexicon')
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[
sample_review_ids]):
    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\finan\AppData\Roaming\nltk_data...

REVIEW: word fail whenever want describe feeling movie sequel flaw sure start
subspecie not execute well enough special effect glorify movie herd movie mas
s consumer care quantity quality cheap fun depth crap like blade not even des
erve capital letter underworlddracula 2000dracula 3000 good movie munch popco
rn drink couple coke make subspecie superior effort anyone claim vampire fana
tic hand obvious vampire romanian story set transylvania scene film location
convince atmosphere not base action pack chase expensive orchestral music rad
u source atmosphere vampire look like behave add breathtakingly gloomy castle
dark passageway situate romania include typical vampiric element movement sha
dow wall vampire take flight work art short like fascinated vampire feel appe
arance well setting sinister dark no good place look subspecie movie vampire
journal brilliant spin former
Actual Sentiment: positive

c:\users\finan\appdata\local\programs\python\python37\lib\site-packages\ipyke
rnel_launcher.py:27: FutureWarning: the 'labels' keyword is deprecated, use
'codes' instead

     SENTIMENT STATS:
  Predicted Sentiment Polarity Score             Positive Negative Neutral
0           positive           0.98  28.000000000000004%    11.0%   61.0%
------------------------------------------------------------
REVIEW: good family movie laugh wish not much school stuff like bully fill mo
vie also seem little easy save piece land build mean flow easily make aware w
ildlife cute way introduce piece land fast runner little slow little hokey re
mind go back school oh dvd chock full goody not miss 7 10 movie 10 10 dvd ext
ra well worth watch well worth time see
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative             Neutral
0           positive           0.97    39.0%     4.0%  57.99999999999999%
------------------------------------------------------------
REVIEW: opinion movie not good hardly find good thing say still would like ex
plain conclude another bad movie decide watch costas mandylor star main reaso
n watch till end like action movie understand movie build action rather story
know not go detail come credibility story event even not explain scene lack s
ense reality look ridiculous beginning movie look quite promising tough good
look specialist not tough smart funny partner must job turn bit different exp
ect story take place cruise ship disaster happen ship turn leave alive strugg
le survive escape shark professional killer rise water furthermore movie quit
e violent main weapon beside disaster already take passenger gun successfully
use many case personally miss good man man woman woman prefer fight family fu
n not think think movie shoot hurry without real vision try say make usual ac
tion movie trick bit something call love without real meaning result bad movi
e
Actual Sentiment: negative
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative             Neutral
0           negative          -0.98    12.0%    31.0%  56.00000000000001%
------------------------------------------------------------
```

Predict sentiment for test dataset

In [17]:
```
predicted_sentiments = [analyze_sentiment_vader_lexicon(review, threshold=0.4,
verbose=False) for review in test_reviews]
```

Evaluate model performance

In [18]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_l
abels=predicted_sentiments,
                                  classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.6964
Precision: 0.704
Recall: 0.6964
F1 Score: 0.6929

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

    positive       0.67      0.80      0.73      7587
    negative       0.74      0.59      0.66      7413

   micro avg       0.70      0.70      0.70     15000
   macro avg       0.70      0.70      0.69     15000
weighted avg       0.70      0.70      0.69     15000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       6066     1521
        negative       3033     4380
```

In [ ]:

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

In [2]:

```
dataset = pd.read_csv(r'movie_reviews_cleaned.csv')

# take a peek at the data
print(dataset.head())
reviews = np.array(dataset['review'])
sentiments = np.array(dataset['sentiment'])

# build train and test datasets
norm_train_reviews = reviews[:35000]
train_sentiments = sentiments[:35000]
norm_test_reviews = reviews[35000:]
test_sentiments = sentiments[35000:]
'''
# normalize datasets
norm_train_reviews = tn.normalize_corpus(train_reviews)
norm_test_reviews = tn.normalize_corpus(test_reviews)
'''
```

```
                                              review sentiment
0  not bother think would see movie great supspen...  negative
1  careful one get mitt change way look kung fu f...  positive
2  chili palmer tired movie know want success mus...  negative
3  follow little know 1998 british film make budg...  positive
4  dark angel cross huxley brave new world percys...  positive
```

Out[2]:

```
'\n# normalize datasets\nnorm_train_reviews = tn.normalize_corpus(train_rev
iews)\nnorm_test_reviews = tn.normalize_corpus(test_reviews)\n'
```

Traditional Supervised Machine Learning Models

Feature Engineering

In [3]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# build BOW features on train reviews
cv = CountVectorizer(binary=False, min_df=0.0, max_df=1.0, ngram_range=(1,2))
cv_train_features = cv.fit_transform(norm_train_reviews)
# build TFIDF features on train reviews
tv = TfidfVectorizer(use_idf=True, min_df=0.0, max_df=1.0, ngram_range=(1,2),
                     sublinear_tf=True)
tv_train_features = tv.fit_transform(norm_train_reviews)
```

In [4]:
```
# transform test reviews into features
cv_test_features = cv.transform(norm_test_reviews)
tv_test_features = tv.transform(norm_test_reviews)
```

In [5]:
```
print('BOW model:> Train features shape:', cv_train_features.shape, ' Test fe
print('TFIDF model:> Train features shape:', tv_train_features.shape, ' Test
```

```
BOW model:> Train features shape: (35000, 2099202)  Test features shape: (1
5000, 2099202)
TFIDF model:> Train features shape: (35000, 2099202)  Test features shape:
(15000, 2099202)
```

Model Training, Prediction and Performance Evaluation

In [6]:
```
from sklearn.linear_model import SGDClassifier, LogisticRegression

lr = LogisticRegression(penalty='l2', max_iter=100, C=1)
svm = SGDClassifier(loss='hinge', n_iter=100)
```

In [7]:

```
# Logistic Regression model on BOW features
lr_bow_predictions = meu.train_predict_model(classifier=lr,
                                             train_features=cv_train_features
                                             test_features=cv_test_features,
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_
                                      classes=['positive', 'negative'])
```

```
c:\users\finan\appdata\local\programs\python\python37\lib\site-packages\skl
earn\linear_model\logistic.py:433: FutureWarning: Default solver will be ch
anged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

Model Performance metrics:
------------------------------
Accuracy: 0.8985
Precision: 0.8986
Recall: 0.8985
F1 Score: 0.8985

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

    positive       0.89      0.91      0.90      7587
    negative       0.90      0.89      0.90      7413

   micro avg       0.90      0.90      0.90     15000
   macro avg       0.90      0.90      0.90     15000
weighted avg       0.90      0.90      0.90     15000


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

C:\Users\finan\Downloads\ML1010_InClass-master\ML1010_InClass-master\Day1\3
_sentiment\model_evaluation_utils.py:62: FutureWarning: the 'labels' keywor
d is deprecated, use 'codes' instead
  labels=level_labels),
C:\Users\finan\Downloads\ML1010_InClass-master\ML1010_InClass-master\Day1\3
_sentiment\model_evaluation_utils.py:64: FutureWarning: the 'labels' keywor
d is deprecated, use 'codes' instead
  labels=level_labels))

                 Predicted:
                   positive negative
Actual: positive       6874      713
        negative        809     6604
```

In [8]:

```
# Logistic Regression model on TF-IDF features
lr_tfidf_predictions = meu.train_predict_model(classifier=lr,
                                               train_features=tv_train_featur
                                               test_features=tv_test_features
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_
                                      classes=['positive', 'negative'])
```

```
F1 Score: 0.8919

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

    positive       0.89      0.90      0.89      7587
    negative       0.90      0.88      0.89      7413

   micro avg       0.89      0.89      0.89     15000
   macro avg       0.89      0.89      0.89     15000
weighted avg       0.89      0.89      0.89     15000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       6852      735
        negative        886     6527
```

In [9]:
```
svm_bow_predictions = meu.train_predict_model(classifier=svm,
                                              train_features=cv_train_features
                                              test_features=cv_test_features,
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_
                                      classes=['positive', 'negative'])
```

```
c:\users\finan\appdata\local\programs\python\python37\lib\site-packages\skl
earn\linear_model\stochastic_gradient.py:152: DeprecationWarning: n_iter pa
rameter is deprecated in 0.19 and will be removed in 0.21. Use max_iter and
tol instead.
  DeprecationWarning)

Model Performance metrics:
------------------------------
Accuracy: 0.8968
Precision: 0.897
Recall: 0.8968
F1 Score: 0.8968

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

    positive       0.89      0.91      0.90      7587
    negative       0.90      0.88      0.89      7413

   micro avg       0.90      0.90      0.90     15000
   macro avg       0.90      0.90      0.90     15000
weighted avg       0.90      0.90      0.90     15000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       6897      690
        negative        858     6555
```

In [10]:
```
svm_tfidf_predictions = meu.train_predict_model(classifier=svm,
                                                train_features=tv_train_featu
                                                test_features=tv_test_feature
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_
                                      classes=['positive', 'negative'])
```

```
c:\users\finan\appdata\local\programs\python\python37\lib\site-packages\skl
earn\linear_model\stochastic_gradient.py:152: DeprecationWarning: n_iter pa
rameter is deprecated in 0.19 and will be removed in 0.21. Use max_iter and
tol instead.
  DeprecationWarning)

Model Performance metrics:
------------------------------
Accuracy: 0.8953
Precision: 0.8956
Recall: 0.8953
F1 Score: 0.8952

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

    positive       0.89      0.91      0.90      7587
    negative       0.91      0.88      0.89      7413

   micro avg       0.90      0.90      0.90     15000
   macro avg       0.90      0.90      0.90     15000
weighted avg       0.90      0.90      0.90     15000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       6912      675
        negative        896     6517
```

Newer Supervised Deep Learning Models

In [11]:
```
import gensim
import keras
from keras.models import Sequential
from keras.layers import Dropout, Activation, Dense
from sklearn.preprocessing import LabelEncoder
```

```
Using TensorFlow backend.
```

Prediction class label encoding

In [12]:

```
le = LabelEncoder()
num_classes=2
# tokenize train reviews & encode train labels
tokenized_train = [tn.tokenizer.tokenize(text)
                   for text in norm_train_reviews]
y_tr = le.fit_transform(train_sentiments)
y_train = keras.utils.to_categorical(y_tr, num_classes)
# tokenize test reviews & encode test labels
tokenized_test = [tn.tokenizer.tokenize(text)
                   for text in norm_test_reviews]
y_ts = le.fit_transform(test_sentiments)
y_test = keras.utils.to_categorical(y_ts, num_classes)
```

In [13]:

```
# print class label encoding map and encoded labels
print('Sentiment class label map:', dict(zip(le.classes_, le.transform(le.cla
print('Sample test label transformation:\n'+'-'*35,
      '\nActual Labels:', test_sentiments[:3], '\nEncoded Labels:', y_ts[:3],
      '\nOne hot encoded Labels:\n', y_test[:3])
```

```
Sentiment class label map: {'negative': 0, 'positive': 1}
Sample test label transformation:
-----------------------------------
Actual Labels: ['negative' 'negative' 'positive']
Encoded Labels: [0 0 1]
One hot encoded Labels:
 [[1. 0.]
 [1. 0.]
 [0. 1.]]
```

Feature Engineering with word embeddings

In [14]:

```
# build word2vec model
w2v_num_features = 500
w2v_model = gensim.models.Word2Vec(tokenized_train, size=w2v_num_features, wi
                                   min_count=10, sample=1e-3)
```

In [15]:
```
def averaged_word2vec_vectorizer(corpus, model, num_features):
    vocabulary = set(model.wv.index2word)

    def average_word_vectors(words, model, vocabulary, num_features):
        feature_vector = np.zeros((num_features,), dtype="float64")
        nwords = 0.

        for word in words:
            if word in vocabulary:
                nwords = nwords + 1.
                feature_vector = np.add(feature_vector, model[word])
        if nwords:
            feature_vector = np.divide(feature_vector, nwords)

        return feature_vector

    features = [average_word_vectors(tokenized_sentence, model, vocabulary, r
                    for tokenized_sentence in corpus]
    return np.array(features)
```

In [16]:
```
# generate averaged word vector features from word2vec model
avg_wv_train_features = averaged_word2vec_vectorizer(corpus=tokenized_train,
                                                     num_features=500)
avg_wv_test_features = averaged_word2vec_vectorizer(corpus=tokenized_test, mo
                                                    num_features=500)
```

```
c:\users\finan\appdata\local\programs\python\python37\lib\site-packages\ipy
kernel_launcher.py:11: DeprecationWarning: Call to deprecated `__getitem__`
(Method will be removed in 4.0.0, use self.wv.__getitem__() instead).
  # This is added back by InteractiveShellApp.init_path()
```

In [17]:
```
# feature engineering with GloVe model
train_nlp = [tn.nlp(item) for item in norm_train_reviews]
train_glove_features = np.array([item.vector for item in train_nlp])

test_nlp = [tn.nlp(item) for item in norm_test_reviews]
test_glove_features = np.array([item.vector for item in test_nlp])
```

In [18]:
```
print('Word2Vec model:> Train features shape:', avg_wv_train_features.shape,
print('GloVe model:> Train features shape:', train_glove_features.shape, ' Te
```

```
Word2Vec model:> Train features shape: (35000, 500)  Test features shape:
(15000, 500)
GloVe model:> Train features shape: (35000, 96)  Test features shape: (1500
0, 96)
```

Modeling with deep neural networks

Building Deep neural network architecture

In [22]:
```
def construct_deepnn_architecture(num_input_features):
    dnn_model = Sequential()
    dnn_model.add(Dense(512, activation='relu', input_shape=(num_input_featur
    dnn_model.add(Dropout(0.2))
    dnn_model.add(Dense(512, activation='relu'))
    dnn_model.add(Dropout(0.2))
    dnn_model.add(Dense(512, activation='relu'))
    dnn_model.add(Dropout(0.2))
    dnn_model.add(Dense(2))
    dnn_model.add(Activation('softmax'))

    dnn_model.compile(loss='categorical_crossentropy', optimizer='adam',
                      metrics=['accuracy'])
    return dnn_model
```

In [23]:
```
w2v_dnn = construct_deepnn_architecture(num_input_features=500)
```

Visualize sample deep architecture

In [40]:
```
'''
import pydot
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

SVG(model_to_dot(w2v_dnn, show_shapes=True, show_layer_names=False,
                 rankdir='TB').create(prog='dot', format='svg'))
'''
```

Out[40]:
```
"\nimport pydot\nfrom IPython.display import SVG\nfrom keras.utils.vis_util
s import model_to_dot\n\nSVG(model_to_dot(w2v_dnn, show_shapes=True, show_l
ayer_names=False, \n                 rankdir='TB').create(prog='dot', forma
t='svg'))\n"
```

Model Training, Prediction and Performance Evaluation

In [26]:

```
batch_size = 100
w2v_dnn.fit(avg_wv_train_features, y_train, epochs=5, batch_size=batch_size,
            shuffle=True, validation_split=0.1, verbose=1)
```

```
WARNING:tensorflow:From c:\users\finan\appdata\local\programs\python\pyth
on37\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32
(from tensorflow.python.ops.math_ops) is deprecated and will be removed i
n a future version.
Instructions for updating:
Use tf.cast instead.
Train on 31500 samples, validate on 3500 samples
Epoch 1/5
31500/31500 [==============================] - ETA: 2:53 - loss: 0.7068 -
acc: 0.520 - ETA: 1:03 - loss: 0.6180 - acc: 0.670 - ETA: 41s - loss: 0.5
761 - acc: 0.708 - ETA: 31s - loss: 0.5461 - acc: 0.72 - ETA: 26s - loss:
0.4949 - acc: 0.76 - ETA: 21s - loss: 0.4632 - acc: 0.78 - ETA: 19s - los
s: 0.4432 - acc: 0.79 - ETA: 17s - loss: 0.4168 - acc: 0.80 - ETA: 16s -
loss: 0.4118 - acc: 0.81 - ETA: 15s - loss: 0.4023 - acc: 0.82 - ETA: 15s
- loss: 0.3986 - acc: 0.82 - ETA: 14s - loss: 0.3922 - acc: 0.82 - ETA: 1
4s - loss: 0.3898 - acc: 0.82 - ETA: 13s - loss: 0.3814 - acc: 0.83 - ET
A: 12s - loss: 0.3747 - acc: 0.83 - ETA: 12s - loss: 0.3707 - acc: 0.83 -
ETA: 11s - loss: 0.3672 - acc: 0.84 - ETA: 11s - loss: 0.3618 - acc: 0.84
- ETA: 11s - loss: 0.3619 - acc: 0.84 - ETA: 10s - loss: 0.3590 - acc: 0.
```

In [41]:

```
y_pred = w2v_dnn.predict_classes(avg_wv_test_features)
predictions = le.inverse_transform(y_pred)
```

In [28]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_
                                      classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.8795
Precision: 0.8817
Recall: 0.8795
F1 Score: 0.8792

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

    positive       0.85      0.92      0.89      7587
    negative       0.91      0.84      0.87      7413

   micro avg       0.88      0.88      0.88     15000
   macro avg       0.88      0.88      0.88     15000
weighted avg       0.88      0.88      0.88     15000


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

C:\Users\finan\Downloads\ML1010_InClass-master\ML1010_InClass-master\Day1\3
_sentiment\model_evaluation_utils.py:62: FutureWarning: the 'labels' keywor
d is deprecated, use 'codes' instead
  labels=level_labels),
C:\Users\finan\Downloads\ML1010_InClass-master\ML1010_InClass-master\Day1\3
_sentiment\model_evaluation_utils.py:64: FutureWarning: the 'labels' keywor
d is deprecated, use 'codes' instead
  labels=level_labels))

                 Predicted:
                   positive negative
Actual: positive       6974      613
        negative       1195     6218
```

In [39]:
```
glove_dnn = construct_deepnn_architecture(num_input_features=96)
```

In [38]:
```
batch_size = 100
glove_dnn.fit(train_glove_features, y_train, epochs=5, batch_size=batch_size,
              shuffle=True, validation_split=0.1, verbose=1)
```

```
c: 0.537 - ETA: 8s - loss: 0.7295 - acc: 0.539 - ETA: 8s - loss: 0.7283 -
acc: 0.537 - ETA: 7s - loss: 0.7258 - acc: 0.540 - ETA: 7s - loss: 0.7241
- acc: 0.542 - ETA: 7s - loss: 0.7230 - acc: 0.542 - ETA: 7s - loss: 0.72
21 - acc: 0.544 - ETA: 7s - loss: 0.7201 - acc: 0.545 - ETA: 7s - loss:
0.7188 - acc: 0.546 - ETA: 7s - loss: 0.7184 - acc: 0.546 - ETA: 6s - los
s: 0.7168 - acc: 0.547 - ETA: 6s - loss: 0.7159 - acc: 0.546 - ETA: 6s -
loss: 0.7146 - acc: 0.548 - ETA: 6s - loss: 0.7135 - acc: 0.549 - ETA: 6s
- loss: 0.7126 - acc: 0.549 - ETA: 6s - loss: 0.7118 - acc: 0.551 - ETA:
6s - loss: 0.7105 - acc: 0.552 - ETA: 5s - loss: 0.7097 - acc: 0.553 - ET
A: 5s - loss: 0.7090 - acc: 0.554 - ETA: 5s - loss: 0.7079 - acc: 0.556 -
ETA: 5s - loss: 0.7069 - acc: 0.557 - ETA: 5s - loss: 0.7059 - acc: 0.557
- ETA: 5s - loss: 0.7047 - acc: 0.558 - ETA: 5s - loss: 0.7042 - acc: 0.5
58 - ETA: 5s - loss: 0.7038 - acc: 0.558 - ETA: 5s - loss: 0.7034 - acc:
0.559 - ETA: 5s - loss: 0.7032 - acc: 0.558 - ETA: 5s - loss: 0.7026 - ac
c: 0.559 - ETA: 5s - loss: 0.7025 - acc: 0.559 - ETA: 5s - loss: 0.7019 -
acc: 0.560 - ETA: 5s - loss: 0.7013 - acc: 0.561 - ETA: 5s - loss: 0.7006
- acc: 0.561 - ETA: 4s - loss: 0.7007 - acc: 0.561 - ETA: 4s - loss: 0.70
02 - acc: 0.561 - ETA: 4s - loss: 0.7001 - acc: 0.561 - ETA: 4s - loss:
0.6996 - acc: 0.561 - ETA: 4s - loss: 0.6993 - acc: 0.562 - ETA: 4s - los
```

In [43]:
```
y_pred = glove_dnn.predict_classes(test_glove_features)
predictions = le.inverse_transform(y_pred)
```

In [44]:

```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_
                                      classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.5055
Precision: 0.4535
Recall: 0.5055
F1 Score: 0.3406

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

    positive       0.51      1.00      0.67      7587
    negative       0.40      0.00      0.00      7413

   micro avg       0.51      0.51      0.51     15000
   macro avg       0.45      0.50      0.34     15000
weighted avg       0.45      0.51      0.34     15000


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive       7575       12
        negative       7405        8
```

In [ ]: