Some Pre-Processing

Import necessary depencencies

In [3]:
```
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 [8]:
```
## Data need clean enough before concuduct
## for test purpose, only apply very samll cleaned data (100 rows, 60/40)


dataset = pd.read_csv(r'yelp_review10Ks_sentiment.csv')


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

# extract data for model evaluation
train_reviews = reviews[:60]
train_sentiments = sentiments[:60]

test_reviews = reviews[60:]
test_sentiments = sentiments[60:]
sample_review_ids = [2, 23, 35]
```

In [9]:
```
# 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=cs
    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[9]:
```
'\nnorm_test_reviews = tn.normalize_corpus(test_reviews)\nnorm_train_reviews =
tn.normalize_corpus(train_reviews)\n#output back to a csv file again\nimport cs
v\nwith open(r\'movie_reviews_cleaned.csv\', mode=\'w\') as cleaned_file:\n
csv_writer = csv.writer(cleaned_file, delimiter=\',\', quotechar=\'"\', quoting
=csv.QUOTE_MINIMAL)\n    csv_writer.writerow([\'review\', \'sentiment\'])\n
for  text, sent in zip(norm_test_reviews, test_sentiments):\n        csv_write
r.writerow([text, sent])\n    for  text, sent in zip(norm_train_reviews, train_
sentiments):\n        csv_writer.writerow([text, sent])\n'
```

In [10]:
```
## Unsupervised (Lexicon)
#-Sentiment Analysis with AFINN
from afinn import Afinn

afn = Afinn(emoticons=True)

# NOTE:  to use afinn score, call the function afn.score("text you want the sentim
# 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 [11]:
```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[samp
    print('Reviews:', review)
    print('Actual Sentiment:', sentiment)
    print('Predicted Sentiment polarity:', afn.score(review))
    print('-'*60)
```

```
Reviews: If could give 4 stars for atmosphere 3 stars for food would

With all hype of getting Margaritaville guess was expecting little more in way
of Margaritas cheeseburgers But was underwhelmed with both u can get much bett
er burger next door at Yardhouse much better selection of Margaritas from Salt
y Senorita With aside place quite spectacle has 2 stories stage fishing boat b
ooths every other piece of decor Parrothead could love TV s broadcast Jim hims
elf rocking out some odd montages of Parrothead outings Live music offered her
e several nights of week they offer 2 3 bar areas defiantly won t become part
of regular rotation but from time to time we might wonder in looking for our l
ost shaker of salt
Actual Sentiment: negative
Predicted Sentiment polarity: 5.0
------------------------------------------------------------
Reviews: Great hotel in Central Phoenix for stay cation but not necessarily pl
ace to stay out of town without car Not much around area unless u re familiar
with downtown would rather have guest stay in Old Town Scottsdale etc BUT u do
stay here s awesome Great boutique rooms Awesome pool s happening in summer GR
EAT rooftop patio bar very very busy lobby with Gallo Blanco attached great pl
ace to stay but have car
Actual Sentiment: Positive
Predicted Sentiment polarity: 18.0
------------------------------------------------------------
Reviews: Awesome subs clean friendly well priced
Actual Sentiment: Positive
Predicted Sentiment polarity: 8.0
------------------------------------------------------------
```

Predict sentiment for test dataset

In [12]:
```
sentiment_polarity = [afn.score(review) for review in test_reviews]
predicted_sentiments = ['positive' if score >= 1.0 else 'negative' for score in se
```

Evaluate model performance

In [13]:
```
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_label
                                  classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.0769

C:\Users\lijua\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1
143: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples.
  'precision', 'predicted', average, warn_for)

Precision: 0.1731

C:\Users\lijua\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1
145: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in lab
els with no true samples.
  'recall', 'true', average, warn_for)

Recall: 0.0769

C:\Users\lijua\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1
143: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in la
bels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\lijua\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1
145: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in la
bels with no true samples.
  'recall', 'true', average, warn_for)

F1 Score: 0.1065

Model Classification report:
------------------------------

C:\Users\lijua\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1
145: UndefinedMetricWarning: Recall and F-score are ill-defined and being set
to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)

              precision    recall  f1-score   support

    positive       0.00      0.00      0.00         0
    negative       0.75      0.33      0.46         9

   micro avg       0.08      0.33      0.12         9
   macro avg       0.38      0.17      0.23         9
weighted avg       0.75      0.33      0.46         9


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive          0        0
        negative          6        3
```

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 [14]:
```
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\lijua\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 [15]:
```
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
                                         norm_neg_score, norm_final_score]],
                                       columns=pd.MultiIndex(levels=[['SENTIMENT S
                                                                     ['Predicted Sentiment
                                                                      'Positive', 'Negativ
                                                                     labels=[[0,0,0,0,0],[
        print(sentiment_frame)

    return final_sentiment
```

Predict sentiment for sample reviews

In [19]:
```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[samp
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    pred = analyze_sentiment_sentiwordnet_lexicon(review, verbose=True)
    print('-'*60)
```

```
REVIEW: have no idea whether there Japanese community in part of town but New
Tokyo would suit one well s rather small but they have wide selection of Japan
ese snacks beverages including new fave beer Asahi Black refrigerated items st
aples etc

This not an Asian market s specifically Japanese so don t expect more than few
crossover items like kimchi being said sometimes s good to specialize lets the
m carry more variety of Japanese products

The cashier was very friendly when finally stopped loading basket long enough
to check out Like first reviewer loaded up with snacks for hotel room along wi
th six pack of Asahi Black of course Oh one more important factor prices were
quite reasonable in line with most ethnic market imports maybe even bit cheape
r than usual
Actual Sentiment: Positive
     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.86     0.09     0.05    0.04
------------------------------------------------------------
REVIEW: Great hotel in Central Phoenix for stay cation but not necessarily pla
ce to stay out of town without car Not much around area unless u re familiar w
ith downtown would rather have guest stay in Old Town Scottsdale etc BUT u do
stay here s awesome Great boutique rooms Awesome pool s happening in summer GR
EAT rooftop patio bar very very busy lobby with Gallo Blanco attached great pl
ace to stay but have car
Actual Sentiment: Positive
     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.88     0.07     0.05    0.02
------------------------------------------------------------
REVIEW: Awesome subs clean friendly well priced
Actual Sentiment: Positive
     SENTIMENT STATS:
  Predicted Sentiment Objectivity Positive Negative Overall
0            positive        0.69     0.25     0.06    0.19
------------------------------------------------------------
```

Predict sentiment for test dataset

Build model

In [20]:
```
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=[['SENTIMENT
                                                                      ['Predicted
                                                                       'Positive',
                                                             labels=[[0,0,0,0,0],
        print(sentiment_frame)

    return final_sentiment
```

In [ ]:
```
...
```