Jeffrey Levesque

jlevesqu@syr.edu | An Evaluation of Sentiment Classification Tools

IST-736: HW1

PRofessor gates

Introduction

**Appendix A:** use vader to perform sentiment analysis

def vader\_analysis(fp='{}/data/sample-sentiment.csv'.format(

Path(\_\_file\_\_).resolve().parents[1]

)):

sentences = pd.read\_csv(fp)['SentimentText']

sid = SentimentIntensityAnalyzer()

result = []

for i, s in enumerate(sentences):

ss = sid.polarity\_scores(s)

scores = []

for k in sorted(ss):

scores.append({k: ss[k]})

result.append({i: scores})

return({'sent': sentences, 'result': result})

**Appendix B:** use sklearn and nltk packages to create naive bayes model.

def nb\_model(fp='{}/data/sample-sentiment.csv'.format(

Path(\_\_file\_\_).resolve().parents[1]

)):

data = pd.read\_csv(fp)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

data['SentimentText'],

data['Sentiment'],

test\_size=0.25

)

count\_vect = CountVectorizer(stop\_words='english')

bow = count\_vect.fit\_transform(X\_train)

tfidf\_transformer = TfidfTransformer()

X\_train\_tfidf = tfidf\_transformer.fit\_transform(bow)

clf = MultinomialNB().fit(X\_train\_tfidf, y\_train)

predictions = []

for item in list(X\_test):

prediction = count\_vect.transform([item])

predictions.append(

clf.predict(tfidf\_transformer.fit\_transform(prediction))

)

return({

'model': clf,

'actual': y\_test,

'predicted': predictions

})

**Appendix C:** compare upload time for provided csv.

def time\_df(fp='{}/data/sample-sentiment.csv'.format(

Path(\_\_file\_\_).resolve().parents[1]

)):

start\_pd = time.time()

df\_pd = pd.read\_csv(fp)

pd\_time = time.time() - start\_pd

start\_np = time.time()

with open(fp, 'r') as f:

data = list(csv.reader(f, delimiter=','))

df\_np = pd.DataFrame(data[1:])

df\_np.columns = data[0]

np\_time = time.time() - start\_np

return({

'pd\_time': pd\_time,

'np\_time': np\_time,

'pd\_size': df\_pd.size,

'np\_size': df\_np.size

})

**Appendix D:** implementation of Appendix A, B, C, with confusion matrix for naïve bayes classifier.

if \_\_name\_\_ == '\_\_main\_\_':

# dataframe benchmark

tdf = time\_df()

print('panda upload time: {}'.format(tdf['pd\_time']))

print('numpy upload time: {}'.format(tdf['np\_time']))

# vader analysis

va = vader\_analysis()

[print('{}\n{}\n\n'.format(x, va['result'][i])) for i,x in enumerate(va['sent'])]

# naive bayes prediction

model = nb\_model()

skplt.metrics.plot\_confusion\_matrix(

model['actual'],

model['predicted']

)

plt.show()