Jeffrey Levesque

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

IST-736: HW1

PRofessor gates

**Introduction**

In the field of text mining, and natural language processing, analytical studies can often be approached using various techniques. Moreover, the general basis of text mining often involves the counting of words. Having numerical counts in nested arrays, allows the transformation of textual data to be operated by various modeling techniques. Analysts often make decisions regarding data preparation through analysis. However, reporting deadlines often require analysts to leverage experience to make appropriate decisions.

One may question, what aspects of a study require careful planning? To generalize, time management is often an important component to any project. The ability to properly manage time, allows more complicated segments (often analysis related) the time needed to derive necessary findings for a meaningful conclusion.

As a general practice data analysts and scientists need to be comfortable with the tools required for data preparation. These include familiarity with efficiencies to ingest data, with knowledge of the “right” tools for the corresponding analysis. Having the wisdom and foresight to accomplish these tasks, allow the analyst to spend more time on the crucial elements that build the report.

**Analysis**

Two different analysis were performed. First a dataframe benchmark was conducted. Pandas were compared to a slightly more verbose methodology. The comparisons measured how fast a dataframe could be created from a csv file. Second, sentiment analysis was conducted using the Vader lexicon[[1]](#footnote-1), as well as a more verbose classification approach. Specifically, nltk was used to split the given data into train and test, which allowed a vector count to be converted to a tfidf matrix for a naïve bayes classification (see Appendix D for associated code).

**Data Preparation**

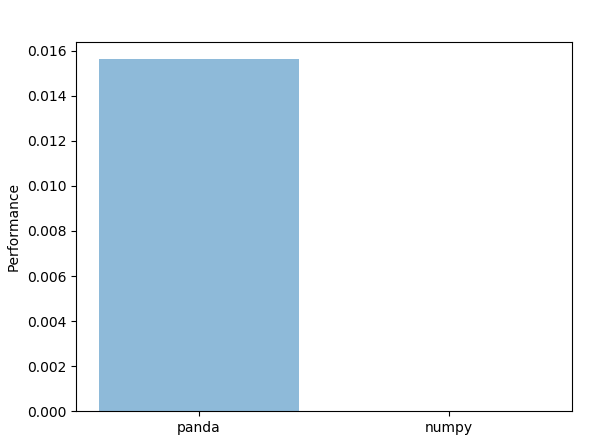
An initial attempt was made to setup a twitter developer account. However, complications including a waiting period for the formal account request, encouraged a manual data collection process. Specifically, using a valid twitter account two terms were searched:

* AI
* AI dangerous

The collected tweets were aggregated into a csv file[[2]](#footnote-2). Specifically, a single column contained two factors - negative sentiment (0), and positive sentiment (1). An attempt was made to keep the distribution relatively balanced. There were no criteria’s defining the selection of tweets, other than generally consisting of more words than hashtags. Therefore, the selection process may be biased. However, 23 instances were positive, while 27 were negative. Another column consisted of the corresponding tweet text.

**Results**

Running the comparison for the dataframe creation indicated that slightly more verbose numpy was slightly faster than the simpler pandas implementation:



**Figure 1:** comparison of upload time using provided csv. See Appendix C and D for associated code.

However, deriving such conclusion is inappropriate. Specifically, running an inferencing methodology, would allow a confidence interval to justify whether the null hypothesis could be rejected. Moreover, implementing a Markov Chain Monte Carlo (MCMC) with a high-density interval (HDI) could show whether a credible difference exists between the two.

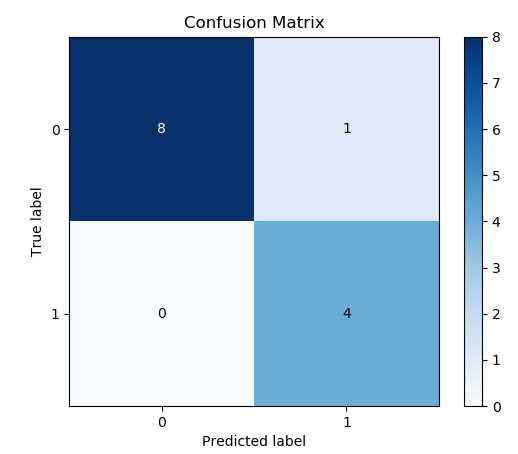
Next, the Vader implementation was applied to provided dataset. The results indicate the percentage likelihood that the provided sentence was negative, neutral, or positive. A full sample output can be reviewed[[3]](#footnote-3). However, the below provides a sample output:

|  |
| --- |
| Join us in San Francisco April 23-25 at the largest event for the Apache Spark and AI community. Use code T200 to save $200! |
| {1: [{'compound': 0.7644}, {'neg': 0.0}, {'neu': 0.734}, {'pos': 0.266}]} |
| The UK is recognised worldwide for its creativity and innovation in #music. Studios like AbbeyRoad are leading the way in sound with the use of #AI in recording. |
| {2: [{'compound': 0.7717}, {'neg': 0.0}, {'neu': 0.765}, {'pos': 0.235}]} |

**Table 1:** example of alternating tweet sentence and sentiment score. Associated code can be reviewed in Appendix A and Appendix D.

Furthermore, since the coding implemented some built-in structures[[4]](#footnote-4), the associated code did not involve splitting the data into a train and test.

Finally, a naïve bayes was trained using 75% of the original data, leaving 25% for testing. Moreover, the sklearn train\_test\_split naturally implement shuffling[[5]](#footnote-5) to randomize splitting. However, this configuration was disabled. Furthermore, since the original data encompassed tweets, the count vectorizer, and tfidf transformer were utilized, along with applying the built-in English stop-words.



**Figure 2:** confusion matrix for naïve bayes classifier on AI tweets. Associated code can be reviewed in Appendix B and Appendix D below.

**Conclusion**

Performing analysis often requires experience and the wisdom to select efficient methodologies. Without applying foundational knowledge for data preparation and associated analysis, could jeopardize time management. Simple approaches such as subsetting the original data, could allow analyst to reduce development time, and focus more on deriving meaningful results. Additionally, many open source communities provide and update many various packages. Keeping abreast of available tools, rather than hardcoding similar approaches, can similarly provide greater time to derive meaningful analysis results.

Unlike the traditional definition of laziness (general lack of motivation), carefully being lazy can often yield great returns. In the field of data analysis, deriving meaningful results is the focal business objective. Thus, automating the necessary tools to derive these end results, are equally important to such conclusions.

**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()

1. <https://www.nltk.org/api/nltk.sentiment.html> [↑](#footnote-ref-1)
2. <https://github.com/jeff1evesque/ist-736-hw/blob/master/data/sample-sentiment.csv> [↑](#footnote-ref-2)
3. <https://github.com/jeff1evesque/ist-736-hw/blob/master/hw1/sample-vader-output.txt> [↑](#footnote-ref-3)
4. <http://www.nltk.org/howto/sentiment.html> [↑](#footnote-ref-4)
5. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html> [↑](#footnote-ref-5)