Predicting Text Corpus Lie Detection with Bernoulli and mNB in Scikit Learn

Brian Hogan

Class: [redacted] Text Mining

Professor Dr. [redacted]

Syracuse University

2019

Introduction

According to Furst, Connors, Bisogni, Sobal, and Falk (1996) "there are many factors involved in complex process of making a single food choice" (Ariyasriwatana & Quiroga, 2016). Online restaurant reviews leave an array of digital footprints and reflections affecting consumer sentiment. The authors surmised there is a large universe of information to help build factors understanding the "ontology of deliciousness." Designing a new factor matrix would help focus data profiling efforts and focus marketing campaigns to increase customer penetration.

Food, food conversations, utensils, and food placement are diverse activities informing the writing of food reviews as a "creativity activity" empowering users with self-improvement and expression (Keuhn (2011) as cited in Ariyasriwatana & Quiroga, 2016). The use of internet site *Yelp* provides a creative platform for harvesting information and perceptions while building a deeper understanding of how words and sentiment influence food perceptions and preferences "often beyond volitional control and sometimes outside conscious awareness" (Chandon and Wansink (2012) as cited in Ariyasriwatana & Quiroga, 2016). This speaks to food advertising as a form of health promotion and broadens the "expectations for the deliciousness and healthiness of [a dining] experience" (Ariyasriwatana & Quiroga, 2016).

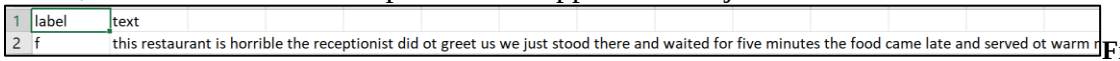
Strategies around prior knowledge can be codified and applied to models to transform text data with both data-driven and concept-driven strategies. Data-driven speaks to vectorizing, association measures, and similarity while concept-driven speaks to abstracting and categorization. Ariyasriwatana and Quiroga's research hypothesized a co-occurrence between discussing cooking and sensory perception as both stimulate how emotions relate to food with "deliciousness… expressed through language as well as nonverbal cues" (2016, p. 18).

The following effort will explore data-driven strategies around positive and negative sentiment labeling of restaurant reviews. Understanding concept-driven strategies will be explored to a lesser degree but using Ariyasriwatana & Quiroga data-driven parameters will help assess if culinary quality, flavor, or unhealthily deliciousness are meaningful facets of labeled restaurant reviews. Finally assessing lie sentiment is performed against test data sets with Bernoulli and Multinomial Naïve Bayes (mNB) algorithms.

Analysis and Models

About the Data:

A collection of 92 restaurant reviews was collected from an Internet collection site similar to Yelp. Reviews were brought into Excel and were labeled with either a "sentiment" or "lie" variable. Referring to Figure 1, text data was joined across multiple columns to create one row per review. A list was generated for either 'lie' or 'sentiment' labeled data and, as discussed below, several vectorization options were applied to the joined data.



	label	text
2	f	this restaurant is horrible the receptionist did ot greet us we just stood there and waited for five minutes the food came late and served ot warm r

Figure 1

Referring to Figure 2, the labeled data is equal in terms of total positive or negative sentiment and true or false lie labels.

Restaurant Raw Data Label Table Counts with Python Seaborn Graphics

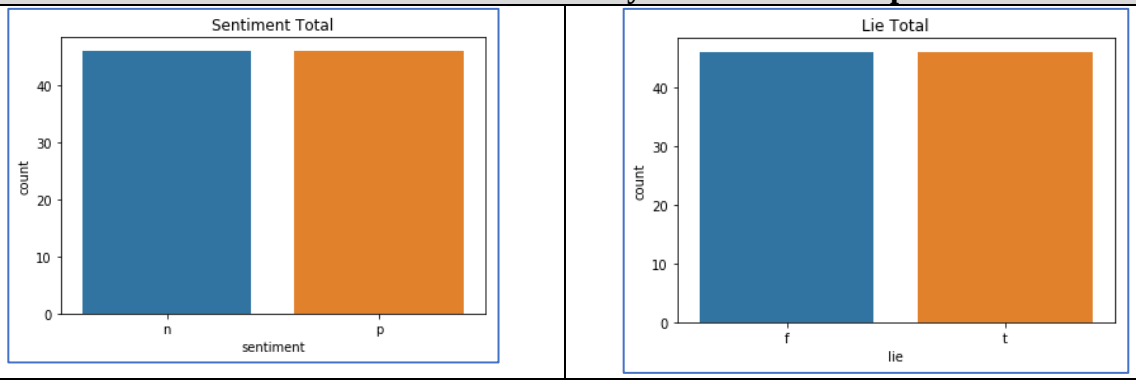


Figure 2

Data Cleaning and Preparation:

Referring to Figure 3, with the assistance of Dr. Gates code "text" data was combined across multiple columns into a single text cell and two files were created based on lie or sentiment labels (Gates, 2019). Data was cleaned for whitespace, annotations, punctuation and lowered. As Sci-Kit is used for vectorization not even small words, i.e. less than three characters, were removed. The cleaning was able to address the label being stripped in the text of the sentence with the Python .pop() command. An index value of 0 removed the single label word with newlist.pop(0). If source data label was >1-character data transformation methods would have been required. Training & testing data were split based on a test size of 0.3 (30%). Data sets were split again at 30% applying Boolean, unigram, and TFIDF vectorization for MNB testing.

The author's Python code is still not as robust as it should be as lie and sentiment data sets are created separately instead of creating in one operation. The only other manual coding component is label creation. Instead of this being done in code reading text index positions [0] and [1] it was separated when the source files were created.

Creation of Source .csv File. Note Row-2 with Pizza correct and other Misc Characters…

Before

lie	sentiment	review
f	n	'Mike\'s Piz NY Service not. Stick to pre-made dishes like stuffed pasta or a s
f	n	'i really like japanese and chinese dishes. we also got a free drink and free
f	n	'After I wer we went to DODO restaurant for dinner. I found worm in one of
f	n	'Olive Oil G and the waitor had no manners whatsoever. Don\'t go to the Ol

After

Label	Text
f	pizza high point y service was very slow and the quality was low you would th they would know at least how to
f	really like this buffet restaurant i marshall street they have a lot of selecti american japanese and chinese dishe
f	i went shopping with some of my friend we went to dodo restaurant for dinner i found worm i one of the dishes

The main imported data file first split by 30% then split again for countvectorization model. NaN values filled.

```
mydata = pd.read_csv('hw6_sentiment_cleaned.csv')
y=train['Label'].values
X=train['Text'].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)
```

Figure 3

Posterior Data Cleaning Curious Learning:

Referring to Figure 4, the current implementation of cleaning is removing any word associated with an apostrophe. It is being grouped together instead of the row being split apart. Another curious finding was finding "whitespace" added to the heading column name "Text." The result was "<space>Text" so data was not read and parsed correctly. This issue was addressed with regular expressions.

Cleaning Issue with Excessive Words Being Removed

	A	B	C
1	lie	sentiment	review
2	f	n	'Mike\'s Piz
3	f	n	'i really like
4	f	n	'After I wer
5	f	n	'Olive Oil G
6	f	n	'The Seven
7	f	n	'I went to X
8	f	n	'I went to A
9	f	n	'I went to t
10	f	n	'OMG. This
11	f	n	'Yesterday
12	f	n	'Last weeke

	A	B
1	Label	Text
2	f	pizza high
3	f	really like
4	f	i went sho
5	f	oil garde v
6	f	seve heave
7	f	went to xy
8	f	went to al
9	f	went to th
10	f	this restau
11	f	i went to a
12	f	weekend i

Figure 4

Finalized Prepped Data & Cleaning Learnings:

Referring to Figure 5, the Python script illustrates data prepared for analysis. Labels were transformed where "0" is (false or negative) and "1" is (true or positive). As per the green highlight, there was an issue with some text being missing or populated with a "0" value requiring removal so it could be included with the Bernoulli algorithm.

Data Cleaning – Addressing NAs

```
mydata = mydata.fillna(0)
from sklearn.model_selection import train_test_split
train, test = train_test_split(mydata, test_size = 0.3)
train.head()
y=train['Label'].values
X=train['Text'].values
#====> this is the second splitting per the training
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
In [239]: print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
(44,) (44,) (20,) (20,)

In [240]: X_train[0]
Out[240]: 'my favorite restaurant yuena restaurant the oodle with beef is
this area i love it so much and so do my friends'

In [241]: y_train[0]
Out[241]: 'f'
```

No lists with NAs for sci-kit

```
impatient that we were wondering whether
it so popular ever be back again',
        0,
        'restaurant rocks! i mea the food
great great just great!!! i love it i l:
        'monkey cafe is my favorite japar
```

Figure 5

Model Family & High-level Operating Parameters:

The following analysis approaches were constructed to build and understanding of how varying vectorization, ngrams, and TF-IDF data transformations would inform Bernoulli binary and mNB models based on coding guidance from Dr. Bei Yu at Syracuse University (Yu, 2019).

Vectorization encoding was performed with “latin-1” and “english” with stop-words removed. There may have been vectorization issues by keeping words with one character found after data cleaning but this observation was not adjusted.

Model Family Tree	
m1-bool-	Boolean unigram with standard “English” stop words removed
m2-mNB-	Unigram with a minimum dataframe of 5, binary=false
m3-mNB-	N-gram range(1,2),minimum dataframe of 5, binary=false
m4-mNB-(tfidf)	A unigram TF-IDF vectorization with a minimum dataframe of 5
m5-bool-	Boolean, ngram_range(1,2), min_df=3
m6-bool-	Boolean, ngram_range(2,3), min_df=4

Figure 6

All models were constructed with Sci-Kit Learn parameters resulting in 400 lines of code to operate the prediction environment. Referring to Figure 7, the first line of code for each model approach as applied to the Lie and Sentiment data. If errors were generated model re-sizing was performed, for example, in experimenting with Boolean data frames were not able to evaluate more than 7 factors.

High-level Model Code Routine Overview

```
m1_X_train_vec = m1_unigram_bool_vectorizer.fit_transform(X_train)
m1_X_test_vec = m1_unigram_bool_vectorizer.transform(X_test)
nb_clf.fit(m1_X_train_vec, y_train)
feature_ranks = sorted(zip(nb_clf.feature_log_prob_[0],m1_unigram_bool_vectorizer.get_feature_names()))
nb_clf.score(m1_X_test_vec, y_test)
m1_y_pred = nb_clf.fit(m1_X_train_vec, y_train).predict(m1_X_test_vec)
m1_cm = confusion_matrix(y_test, m1_y_pred, labels=[0,1])
print(classification_report(y_test, m1_y_pred, target_names=target_names))
posterior_probs = nb_clf.predict_proba(X_test_vec)
nb_clf_pipe = Pipeline([('vect',CountVectorizer(encoding='latin-1',binary=False)),('nb', MultinomialNB())])
```

Figure 7

Results

Results – Positive & Negative Features:

Referring to Figure 8, the inspection of the positive and negative word features informed when a model was correctly or incorrectly generating. Initially, the Boolean transformations were working fine and then seemed not be working with feature ranking. Referring to Figure 9, the data frames are repaired but do illustrate undesired words (green highlighted). This confirms continued learning and execution of proper text data mining cleaning is absolutely necessary to help prediction models excel and calculate correctly.

Assessing Top Word Features – Lie Data (model order left to right from r1c1: m1-m6)	
Positive Features	**Negative Features**
[(-3.674394615657426, 'like'), (-3.6372416288496705, 'menu'), (-3.5658898460801707, 'good'), (-3.550979411226187, 'place'), (-3.526283090214985, 'best'), (-3.3790027806907936, 'went'), (-3.3190081188060176, 'ot'), (-3.2825330532149892, 'food'), (-3.271066157053569, 'whe'), (-3.0464314652548197, 'restaurant')]	[(0.0, 'agai')]
[(0.0, 'amazing')]	[(0.0, 'amazing')]
[(0.0, 'amazing')]	[(0.0, 'amazing')]

Figure 8

Code Correction Generating Properly Ranked Tables

```
In [501]: print(feature_ranks[-10:])
[(-3.666761648860317, 'came'), (-3.5489786132039334, 'america'),
(-3.5489786132039334, 'cooked'), (-3.443618097546107, 'bit'),
(-3.443618097546107, 'case'), (-3.443618097546107, 'dinner'),
(-3.3483079177417823, 'cheese'), (-3.2612965407521526, 'dirty'),
(-2.973614468300372, 'bega'), (-2.9129898464839368, 'coffee')]

In [502]: feature_ranks =
sorted(zip(nb_clf.feature_log_prob_[0],m2_unigram_count_vectorizer.get_feat
ure_names()))

In [503]: print(feature_ranks[-10:])
[(-3.666761648860317, 'ordered'), (-3.5489786132039334, 'best'),
(-3.5489786132039334, 'service'), (-3.443618097546107, 'good'),
(-3.443618097546107, 'ot'), (-3.443618097546107, 'went'),
(-3.3483079177417823, 'place'), (-3.2612965407521526, 'whe'),
(-2.973614468300372, 'food'), (-2.9129898464839368, 'restaurant')]

In [504]: feature_ranks =
sorted(zip(nb_clf.feature_log_prob_[0],m3_gram12_count_vectorizer.get_featu
re_names()))

In [505]: print(feature_ranks[-10:])
[(-3.666761648860317, 'ordered'), (-3.5489786132039334, 'best'),
(-3.5489786132039334, 'service'), (-3.443618097546107, 'good'),
(-3.443618097546107, 'ot'), (-3.443618097546107, 'went'),
(-3.3483079177417823, 'place'), (-3.2612965407521526, 'whe'),
(-2.973614468300372, 'food'), (-2.9129898464839368, 'restaurant')]
```

Figure 9

Lie Results:

Referring to Figure 10, the parameters for model-1 generated the most diverse outcome and correctly identifying a negative label in the test data. None of the other models were able to identify negative sentiments. Does this indicate Boolean is most important? No because there remain underlying issues in data quality limiting predictive quality.

Confusion Matrix → Lie Data

```
In [572]: m5_cm = confusion_matrix(y_test, m5_y_pred, labels=[0,1])

In [573]: m6_cm = confusion_matrix(y_test, m6_y_pred, labels=[0,1])

In [574]: print(m1_cm)
[[14  1]
 [ 3  1]]

In [575]: print(m2_cm)
[[13  2]
 [ 4  0]]

In [576]: print(m3_cm)
[[13  2]
 [ 4  0]]

In [577]: print(m4_cm)
[[15  0]
 [ 4  0]]

In [578]: print(m5_cm)
[[14  1]
 [ 4  0]]

In [579]: print(m6_cm)
[[15  0]
 [ 4  0]]
```

Figure 10

Referring to Figure 11, precision and recall measures overall support prediction quality of the "lie" labels with model 1 parameters working most effectively. In-depth error analysis would help identify mistakenly predicted reviews to help assess strategies for optimizing data cleaning and vectorization.

Classification Reporting → Lie Data – Model Order of Go: row1: m1 & m2; row2: m3 & m4, etc

	precision	recall	f1-score	support
0	0.82	0.93	0.87	15
1	0.50	0.25	0.33	4
micro avg	0.79	0.79	0.79	19
macro avg	0.66	0.59	0.60	19
weighted avg	0.76	0.79	0.76	19

	precision	recall	f1-score	support
0	0.76	0.87	0.81	15
1	0.00	0.00	0.00	4
micro avg	0.68	0.68	0.68	19
macro avg	0.38	0.43	0.41	19
weighted avg	0.60	0.68	0.64	19

	precision	recall	f1-score	support
0	0.76	0.87	0.81	15
1	0.00	0.00	0.00	4
micro avg	0.68	0.68	0.68	19
macro avg	0.38	0.43	0.41	19
weighted avg	0.60	0.68	0.64	19

	precision	recall	f1-score	support
0	0.79	1.00	0.88	15
1	0.00	0.00	0.00	4
micro avg	0.79	0.79	0.79	19
macro avg	0.39	0.50	0.44	19
weighted avg	0.62	0.79	0.70	19

	precision	recall	f1-score	support
0	0.78	0.93	0.85	15
1	0.00	0.00	0.00	4
micro avg	0.74	0.74	0.74	19
macro avg	0.39	0.47	0.42	19
weighted avg	0.61	0.74	0.67	19

	precision	recall	f1-score	support
0	0.79	1.00	0.88	15
1	0.00	0.00	0.00	4
micro avg	0.79	0.79	0.79	19
macro avg	0.39	0.50	0.44	19
weighted avg	0.62	0.79	0.70	19

Figure 11

Referring to Figure 12, further analysis Scikit-Learn's BernoulliNB versus running MultinomialNB didn't reveal anything extraordinary in terms of sentence label prediction but provides an indication of labeling soundness with either approach in the orange predicted labels.

Classification Reporting → Lie Data – Model Order of Go: row1: m1 & m2; row2: m3 & m4, etc

```
#Step 18 -------Brnoulli NB------
from sklearn.naive_bayes import BernoulliNB
m1_X_train_vec_bool = m1_unigram_bool_vectorizer.fit_transform(X_train)
m5_X_train_vec_bool = m5_bool.fit_transform(X_train)
m6_X_train_vec_bool = m6_bool.fit_transform(X_train)
BernoulliNB(m1_X_train_vec_bool, y_train)
BernoulliNB(m5_X_train_vec_bool, y_train)
BernoulliNB(m6_X_train_vec_bool, y_train)
#-----------------------
MultinomialNB(m1_X_train_vec_bool, y_train)
MultinomialNB(m5_X_train_vec_bool, y_train)
MultinomialNB(m6_X_train_vec_bool, y_train)
```

```
Out[604]:
BernoulliNB(alpha=<44x235 sparse matrix of type '<class 'numpy.int64'>'
    with 887 stored elements in Compressed Sparse Row format>,
      binarize=array([0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0],
      dtype=int64),
      class_prior=None, fit_prior=True)

In [605]: MultinomialNB(m1_X_train_vec_bool, y_train)
Out[605]:
MultinomialNB(alpha=<44x235 sparse matrix of type '<class 'numpy.int64'>'
    with 887 stored elements in Compressed Sparse Row format>,
       class_prior=None,
       fit_prior=array([0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0],
      dtype=int64))
```

Figure 12

Referring to Figure 13, cross validation on the lie labels does generate a less reliable outcome compared to precision and recall data. For model 1 approach, average scores were comparable at 0.65 vs 0.67 (green).

Cross validation → Lie Data

```
In [608]: nb_clf_pipe = Pipeline([('vect',CountVectorizer(encoding='latin-1',binary=False)),
('nb', MultinomialNB())])

In [609]: scores = cross_val_score(nb_clf_pipe, X, y, cv=3)

In [610]: mNB_avg=sum(scores)/len(scores)

In [611]: print(mNB_avg)
0.6518037518037517

In [612]: #CROSS VALIDATION

In [613]: nb_clf_pipe = Pipeline([('vect',CountVectorizer(encoding='latin-1',binary=True)),
('nb', MultinomialNB())])

In [614]: scores = cross_val_score(nb_clf_pipe, X, y, cv=3)

In [615]: Bernoulli_avg=sum(scores)/len(scores)

In [616]: print(Bernoulli_avg)
0.6676767676767676
```

Figure 13

Sentiment Results:

How are “sentiment” ratings different from “lie” ratings? To assess this it is important to understand the use of language any anaphora. Whether a review has a positive “lie” rating or not does not necessitate a data coder’s preference for perceived sentiment. This was addressed by each coder having 50/50 split on category labeling in an attempt to generate the most objective data set. It would be less optimal to assess language of specific reviews as individual coders should not be causing bias in the dataset. The following compares sentiment label outcomes between Boolean and multinomial naïve Bayes prediction methods. In either case there is no advantage in using either algorithm.

Row Labels	Count of lie	Count of sentiment
f	46	46
n	23	23
p	23	23
t	46	46
n	23	23
p	23	23
Grand Total	92	92

Figure 14

Referring to Figure 15, sentiment label review rankings are provided in an attempt to see if there any marked differences in words but no difference was ascertained from this approach.

Sentiment Ranking (on left as compared to lie rankings on right)

```
In [664]: print(feature_ranks[-10:])
[(-3.7328963395307104, 'compared'), (-3.550574782736756, 'bit'), (-3.550574782736756,
'cheap'), (-3.550574782736756, 'chinese'), (-3.550574782736756, 'clean'),
(-3.550574782736756, 'didnt'), (-3.3964241029094975, 'champagne'), (-3.1451096746285914,
'did'), (-3.039749158970765, 'come'), (-2.9444389791664403, 'beer')]

In [665]: feature_ranks =
sorted(zip(nb_clf.feature_log_prob_[0],m2_unigram_count_vectorizer.get_feature_names()))

In [666]: print(feature_ranks[-10:])
[(-3.7328963395307104, 'service'), (-3.550574782736756, 'good'), (-3.550574782736756, 'ot'),
(-3.550574782736756, 'place'), (-3.550574782736756, 'really'), (-3.550574782736756, 'whe'),
(-3.3964241029094975, 'ordered'), (-3.1451096746285914, 'went'), (-3.039749158970765,
'restaurant'), (-2.9444389791664403, 'food')]

In [667]: feature_ranks =
sorted(zip(nb_clf.feature_log_prob_[0],m3_gram12_count_vectorizer.get_feature_names()))

In [668]: print(feature_ranks[-10:])
[(-3.7328963395307104, 'service'), (-3.550574782736756, 'good'), (-3.550574782736756, 'ot'),
(-3.550574782736756, 'place'), (-3.550574782736756, 'really'), (-3.550574782736756, 'whe'),
(-3.3964241029094975, 'ordered'), (-3.1451096746285914, 'went'), (-3.039749158970765,
'restaurant'), (-2.9444389791664403, 'food')]

In [669]: feature_ranks =
sorted(zip(nb_clf.feature_log_prob_[0],m4_unigram_tfidf_vectorizer.get_feature_names()))

In [670]: print(feature_ranks[-10:])
[(-3.7328963395307104, 'service'), (-3.550574782736756, 'good'), (-3.550574782736756, 'ot'),
(-3.550574782736756, 'place'), (-3.550574782736756, 'really'), (-3.550574782736756, 'whe'),
(-3.3964241029094975, 'ordered'), (-3.1451096746285914, 'went'), (-3.039749158970765,
'restaurant'), (-2.9444389791664403, 'food')]
```

```
In [501]: print(feature_ranks[-10:])
[(-3.666761648860317, 'came'), (-3.5489786132039334, 'america'),
(-3.5489786132039334, 'cooked'), (-3.443618097546107, 'bit'),
(-3.443618097546107, 'case'), (-3.443618097546107, 'dinner'),
(-3.3483079177417823, 'cheese'), (-3.2612965407521526, 'dirty'),
(-2.973614468300372, 'bega'), (-2.9129898464839368, 'coffee')]

In [502]: feature_ranks =
sorted(zip(nb_clf.feature_log_prob_[0],m2_unigram_count_vectorizer.get_fe
ure_names()))

In [503]: print(feature_ranks[-10:])
[(-3.666761648860317, 'ordered'), (-3.5489786132039334, 'best'),
(-3.5489786132039334, 'service'), (-3.443618097546107, 'good'),
(-3.443618097546107, 'ot'), (-3.443618097546107, 'went'),
(-3.3483079177417823, 'place'), (-3.2612965407521526, 'whe'),
(-2.973614468300372, 'food'), (-2.9129898464839368, 'restaurant')]

In [504]: feature_ranks =
sorted(zip(nb_clf.feature_log_prob_[0],m3_gram12_count_vectorizer.get_fea
re_names()))

In [505]: print(feature_ranks[-10:])
[(-3.666761648860317, 'ordered'), (-3.5489786132039334, 'best'),
(-3.5489786132039334, 'service'), (-3.443618097546107, 'good'),
(-3.443618097546107, 'ot'), (-3.443618097546107, 'went'),
(-3.3483079177417823, 'place'), (-3.2612965407521526, 'whe'),
(-2.973614468300372, 'food'), (-2.9129898464839368, 'restaurant')]
```

Figure 15

Referring to Figure 16, prediction outcomes of sentiment data is vastly different than the lie data. Model unigram factors and vectorization parameters are clearly making an enormous difference. Detailed assessment of model pros and cons primarily between Boolean (red models 1, 5, & 6) and MNB classifier are quite clear in prediction outcomes. Overall all models perform less optimal than the lie models even though the same human label data coders were used.

Confusion Matrix→ Sentiment Data

Sentiment

```
In [703]: m6_cm = confusion_matrix(y_test, m6_y_pred, labels=[0,1])

In [704]: print(m1_cm)
[[1 5]
 [4 9]]

In [705]: print(m2_cm)
[[ 2  4]
 [ 3 10]]

In [706]: print(m3_cm)
[[ 2  4]
 [ 3 10]]

In [707]: print(m4_cm)
[[ 1  5]
 [ 2 11]]

In [708]: print(m5_cm)
[[2 4]
 [4 9]]

In [709]: print(m6_cm)
[[2 4]
 [4 9]]
```

Lie

```
In [572]: m5_cm = confusion_matrix(y_test, m5_y_pred, labels=[0,1])

In [573]: m6_cm = confusion_matrix(y_test, m6_y_pred, labels=[0,1])

In [574]: print(m1_cm)
[[14  1]
 [ 3  1]]

In [575]: print(m2_cm)
[[13  2]
 [ 4  0]]

In [576]: print(m3_cm)
[[13  2]
 [ 4  0]]

In [577]: print(m4_cm)
[[15  0]
 [ 4  0]]

In [578]: print(m5_cm)
[[14  1]
 [ 4  0]]

In [579]: print(m6_cm)
[[15  0]
 [ 4  0]]
```

Figure 16

Referring to Figure 17, overall precision and recall is low but surprising similar based on Boolean or MNB encoding (highlighted green) with m3 (top right) similar to Boolean outcomes in models 5 & 6 (in blue).

Classification Reporting ➔ Sentiment Data

```
In [710]: print(classification_report(y_test, m1_y_pred, t
              precision    recall  f1-score   support

           0       0.20      0.17      0.18         6
           1       0.64      0.69      0.67        13

   micro avg       0.53      0.53      0.53        19
   macro avg       0.42      0.43      0.42        19
weighted avg       0.50      0.53      0.51        19


In [711]: print(classification_report(y_test, m2_y_pred, t
              precision    recall  f1-score   support

           0       0.40      0.33      0.36         6
           1       0.71      0.77      0.74        13

   micro avg       0.63      0.63      0.63        19
   macro avg       0.56      0.55      0.55        19
weighted avg       0.62      0.63      0.62        19


In [712]: print(classification_report(y_test, m3_y_pred, t
              precision    recall  f1-score   support

           0       0.40      0.33      0.36         6
           1       0.71      0.77      0.74        13

   micro avg       0.63      0.63      0.63        19
   macro avg       0.56      0.55      0.55        19
weighted avg       0.62      0.63      0.62        19
```

```
In [713]: print(classification_report(y_test, m4_y_pred
              precision    recall  f1-score   support

           0       0.33      0.17      0.22         6
           1       0.69      0.85      0.76        13

   micro avg       0.63      0.63      0.63        19
   macro avg       0.51      0.51      0.49        19
weighted avg       0.58      0.63      0.59        19


In [714]: print(classification_report(y_test, m5_y_pred
              precision    recall  f1-score   support

           0       0.33      0.33      0.33         6
           1       0.69      0.69      0.69        13

   micro avg       0.58      0.58      0.58        19
   macro avg       0.51      0.51      0.51        19
weighted avg       0.58      0.58      0.58        19


In [715]: print(classification_report(y_test, m6_y_pred
              precision    recall  f1-score   support

           0       0.33      0.33      0.33         6
           1       0.69      0.69      0.69        13

   micro avg       0.58      0.58      0.58        19
   macro avg       0.51      0.51      0.51        19
weighted avg       0.58      0.58      0.58        19
```

Figure 17

Referring to Figure 18, cross validation reveals a Bernoulli approach is more favorable. This finding wasn't determined with lie coded data.

Cross validation → Sentiment Data

```
In [716]: nb_clf_pipe = Pipeline([('vect',CountVect
('nb', MultinomialNB())])

In [717]: scores = cross_val_score(nb_clf_pipe, X,

In [718]: mNB_avg=sum(scores)/len(scores)

In [719]: print(mNB_avg)
0.6190476190476191

In [720]: nb_clf_pipe = Pipeline([('vect',CountVect
('nb', MultinomialNB())])

In [721]: scores = cross_val_score(nb_clf_pipe, X,

In [722]: Bernoulli_avg=sum(scores)/len(scores)

In [723]: print(Bernoulli_avg)
0.6507936507936508
```

Figure 18

Conclusion

Ariyasriwatana & Quiroga's research focused on using 205 online reviews of 41 restaurants building a coding matrix mapping words to context categories such as smells, flavor, culinary affairs, and other novel categories such as "delicious" to help assess healthy or unhealthy opinions. An associated qualitative analysis approach helped build unique categories and other meaningful analysis parameters.

Any analysis process can be improved by adding a predictive component to assist with data categorization. With the help of Sci-Kit text mining features, models can be trained to assist sentiment labeling algorithms to understand, compare, and categorize data. This analysis was unsuccessful assessing lie categories in the restaurant review data making it difficult to validate Ariyasriwatana & Quiroga's approach. The use of only "lie and sentiment" labels perhaps could have been improved prediction results by marking more distinct review label subtypes such as "obvious lie" and "not so obvious lie" to build more label prediction distinction.

Clearly the creation of higher-level data bucket categories has merit and could be useful for helping future data miners improve social media "filtering." Any advancements in filtering can assist advertisers or business owners with connecting to individual preferences, such as a healthy or fast food restaurant, and better informing an existing or new customer with potentially robust restaurant portfolio choices.

References

Ariyasriwatana, W., Quiroga, L (2016). A thousand ways to say ‘Delicious!’ – Categorizing expressions of deliciousness from restaurant reviews on the social network site Yelp. Appetite. Retrieved from: http://dx.doi.org/10.1016/j.appet.2016.01.002

[redacted]., 2019. Python and Text Mining Lecture Notes. Syracuse University, [redacted].