*You are currently looking at **version 1.0** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the [Jupyter Notebook FAQ (https://www.coursera.org/learn/python-text-mining/resources/d9pwm)](https://www.coursera.org/learn/python-text-mining/resources/d9pwm) course resource.*

Note: Some of the cells in this notebook are computationally expensive. To reduce runtime, this notebook is using a subset of the data.

Case Study: Sentiment Analysis

Data Prep

In [1]:

```
import pandas as pd
import numpy as np

# Read in the data
df = pd.read_csv('Amazon_Unlocked_Mobile.csv')

# Sample the data to speed up computation
# Comment out this line to match with lecture
df = df.sample(frac=0.1, random_state=10)

df.head()
```

Out[1]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
394349	Sony XPERIA Z2 D6503 FACTORY UNLOCKED Internat...	NaN	244.95	5	Very good one! Better than Samsung S and iphon...	0.0
34377	Apple iPhone 5c 8GB (Pink) - Verizon Wireless	Apple	194.99	1	The phone needed a SIM card, would have been n...	1.0
248521	Motorola Droid RAZR MAXX XT912 M Verizon Smart...	Motorola	174.99	5	I was 3 months away from my upgrade and my Str...	3.0
167661	CNPGD [U.S. Office Extended Warranty] Smartwat...	CNPGD	49.99	1	an experience i want to forget	0.0
73287	Apple iPhone 7 Unlocked Phone 256 GB - US Vers...	Apple	922.00	5	GREAT PHONE WORK ACCORDING MY EXPECTATIONS.	1.0

In [2]:

```
# Drop missing values
df.dropna(inplace=True)

# Remove any 'neutral' ratings equal to 3
df = df[df['Rating'] != 3]

# Encode 4s and 5s as 1 (rated positively)
# Encode 1s and 2s as 0 (rated poorly)
df['Positively Rated'] = np.where(df['Rating'] > 3, 1, 0)
df.head(10)
```

Out[2]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Positively Rated
34377	Apple iPhone 5c 8GB (Pink) - Verizon Wireless	Apple	194.99	1	The phone needed a SIM card, would have been n...	1.0	0
248521	Motorola Droid RAZR MAXX XT912 M Verizon Smart...	Motorola	174.99	5	I was 3 months away from my upgrade and my Str...	3.0	1
167661	CNPGD [U.S. Office Extended Warranty] Smartwat...	CNPGD	49.99	1	an experience i want to forget	0.0	0
73287	Apple iPhone 7 Unlocked Phone 256 GB - US Vers...	Apple	922.00	5	GREAT PHONE WORK ACCORDING MY EXPECTATIONS.	1.0	1
277158	Nokia N8 Unlocked GSM Touch Screen Phone Featu...	Nokia	95.00	5	I fell in love with this phone because it did ...	0.0	1
100311	Blackberry Torch 2 9810 Unlocked Phone with 1....	BlackBerry	77.49	5	I am pleased with this Blackberry phone! The p...	0.0	1
251669	Motorola Moto E (1st Generation) - Black - 4 G...	Motorola	89.99	5	Great product, best value for money smartphone...	0.0	1
279878	OtterBox 77-29864 Defender Series Hybrid Case ...	OtterBox	9.99	5	I've bought 3 no problems. Fast delivery.	0.0	1
406017	Verizon HTC Rezound 4G Android Smarphone - 8MP...	HTC	74.99	4	Great phone for the price...	0.0	1
302567	RCA M1 Unlocked Cell Phone, Dual Sim, 5Mp Came...	RCA	159.99	5	My mom is not good with new technoloy but this...	4.0	1

In [3]:

```
# Most ratings are positive
df['Positively Rated'].mean()
```

Out[3]:

0.7471776686078667

In [4]:

```
from sklearn.model_selection import train_test_split

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df['Reviews'],
                                                    df['Positively Rated'],
                                                    random_state=0)
```

In [5]:

```
print('X_train first entry:\n\n', X_train.iloc[0])
print('\n\nX_train shape: ', X_train.shape)
```

```
X_train first entry:

 Everything about it is awesome!


X_train shape:  (23052,)
```

CountVectorizer

In [6]:

```
from sklearn.feature_extraction.text import CountVectorizer

# Fit the CountVectorizer to the training data
vect = CountVectorizer().fit(X_train)
```

In [7]:

```
vect.get_feature_names()[::2000]
```

Out[7]:

```
['00',
 'arroja',
 'comapañias',
 'dvds',
 'golden',
 'lands',
 'oil',
 'razonable',
 'smallsliver',
 'tweak']
```

In [8]:

```
len(vect.get_feature_names())
```

Out[8]:

```
19601
```

In [9]:

```
# transform the documents in the training data to a document-term matrix
X_train_vectorized = vect.transform(X_train)

X_train_vectorized
```

Out[9]:

```
<23052x19601 sparse matrix of type '<class 'numpy.int64'>'
	with 613289 stored elements in Compressed Sparse Row format>
```

In [10]:

```
from sklearn.linear_model import LogisticRegression

# Train the model
model = LogisticRegression()
model.fit(X_train_vectorized, y_train)
```

Out[10]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [11]:

```
from sklearn.metrics import roc_auc_score

# Predict the transformed test documents
predictions = model.predict(vect.transform(X_test))

# Any word in X_test that didn't appear in X_train will just be ignored.
print('AUC: ', roc_auc_score(y_test, predictions))
```

```
AUC:  0.897433277667
```

In [12]:

```
# get the feature names as numpy array
feature_names = np.array(vect.get_feature_names())

# Sort the coefficients from the model
sorted_coef_index = model.coef_[0].argsort()

# Find the 10 smallest and 10 largest coefficients
# The 10 largest coefficients are being indexed using [:-11:-1]
# so the list returned is in order of largest to smallest
print('Smallest Coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest Coefs: \n{}'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest Coefs:
['worst' 'terrible' 'slow' 'junk' 'poor' 'sucks' 'horrible' 'useless'
 'waste' 'disappointed']

Largest Coefs:
['excelent' 'excelente' 'excellent' 'perfectly' 'love' 'perfect' 'exactly'
 'great' 'best' 'awesome']
```

Tfidf

In [13]:

```
# Term frequency inverse document frequency
from sklearn.feature_extraction.text import TfidfVectorizer

# Fit the TfidfVectorizer to the training data specifiying a minimum document frequency of
vect = TfidfVectorizer(min_df=5).fit(X_train)
len(vect.get_feature_names())
```

Out[13]:

```
5442
```

In [14]:

```
X_train_vectorized = vect.transform(X_train)

model = LogisticRegression()
model.fit(X_train_vectorized, y_train)

predictions = model.predict(vect.transform(X_test))

print('AUC: ', roc_auc_score(y_test, predictions))
```

```
AUC:  0.889951006492
```

In [15]:

```
feature_names = np.array(vect.get_feature_names())

sorted_tfidf_index = X_train_vectorized.max(0).toarray()[0].argsort()

print('Smallest tfidf:\n{}\n'.format(feature_names[sorted_tfidf_index[:10]]))
print('Largest tfidf: \n{}'.format(feature_names[sorted_tfidf_index[:-11:-1]]))
```

```
Smallest tfidf:
['61' 'printer' 'approach' 'adjustment' 'consequences' 'length' 'emailing'
 'degrees' 'handsfree' 'chipset']

Largest tfidf:
['unlocked' 'handy' 'useless' 'cheat' 'up' 'original' 'exelent' 'exelente'
 'exellent' 'satisfied']
```

In [16]:

```
sorted_coef_index = model.coef_[0].argsort()

print('Smallest Coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest Coefs: \n{}'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest Coefs:
['not' 'slow' 'disappointed' 'worst' 'terrible' 'never' 'return' 'doesn'
 'horrible' 'waste']

Largest Coefs:
['great' 'love' 'excellent' 'good' 'best' 'perfect' 'price' 'awesome' 'far'
 'perfectly']
```

In [17]:

```
# These reviews are treated the same by our current model
print(model.predict(vect.transform(['not an issue, phone is working',
                                    'an issue, phone is not working'])))
```

```
[0 0]
```

n-grams

In [18]:

```
# Fit the CountVectorizer to the training data specifiying a minimum
# document frequency of 5 and extracting 1-grams and 2-grams
vect = CountVectorizer(min_df=5, ngram_range=(1,2)).fit(X_train)

X_train_vectorized = vect.transform(X_train)

len(vect.get_feature_names())
```

Out[18]:

```
29072
```

In [19]:

```
model = LogisticRegression()
model.fit(X_train_vectorized, y_train)

predictions = model.predict(vect.transform(X_test))

print('AUC: ', roc_auc_score(y_test, predictions))
```

```
AUC:  0.91106617946
```

In [20]:

```
feature_names = np.array(vect.get_feature_names())

sorted_coef_index = model.coef_[0].argsort()

print('Smallest Coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest Coefs: \n{}'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest Coefs:
['no good' 'junk' 'poor' 'slow' 'worst' 'broken' 'not good' 'terrible'
 'defective' 'horrible']

Largest Coefs:
['excellent' 'excelente' 'excelent' 'perfect' 'great' 'love' 'awesome'
 'no problems' 'good' 'best']
```

In [21]:

```
# These reviews are now correctly identified
print(model.predict(vect.transform(['not an issue, phone is working',
                                    'an issue, phone is not working'])))
```

```
[1 0]
```