In [1]:

```
import numpy as np
import nltk
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as st
```

In [2]:

```
news_csv = pd.read_csv("news_data/news_reuters_10.csv", error_bad_lines=Fal
se, header = None, names = ["stock", "company", "date", "title", "summary",
"type", "website"])
google_price_csv = pd.read_csv("price_data/GOOGL_2006-01-01_to_2017-11-01.c
sv")
```

In [3]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
```

In [8]:

```
sid.polarity_scores("im gay")
```

Out[8]:

```
{'compound': 0.0, 'neg': 0.0, 'neu': 1.0, 'pos': 0.0}
```

In [4]:

```
number_to_month = {"01": "Jan", "02":"Feb", "03":"Mar", "04":"Apr", "05":"M
ay", "06": "Jun", "07":"Jul", "08":"Aug", "09":"Sep", "10":"Oct", "11":"Nov
", "12":"Dec"}
def conv_num_to_string(d):
    year = d[0:4]
    month = d[4:6]
    day = d[6:8]
    new = day + "-" + number_to_month[month] + "-" + year[2:4]
    return new
```

In [5]:

```
def up_down_ratio(stock, day_lag): #ex: sentiment_to_price_plot("AAPL", 1,
'neg')
    stock_data = news_csv[news_csv["stock"] == stock]
    stock_price_csv = pd.read_csv("price_data/"+ stock+"_2006-01-
01_to_2017-11-01.csv")
    total = []
    for index, row in stock_data.iterrows():


        day = conv_num_to_string(str(row["date"]) )

        if day in stock_price_csv["Date"].values:
```

```
            row_index = stock_price_csv.index[stock_price_csv["Date"] == da
].tolist()[0]
            next_price = stock_price_csv.iloc[row_index - day_lag  ]
            #print next_price["Date"], google_price_csv.iloc[row_index]["Da
te"]
            diff = next_price["Close"] - next_price["Open"]
            if diff >= 0.0:
                total.append(1)
            else:
                total.append(0)
    return 100*sum(total)/len(total)
```

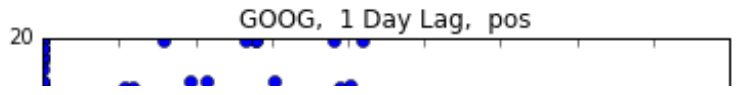
In [6]:

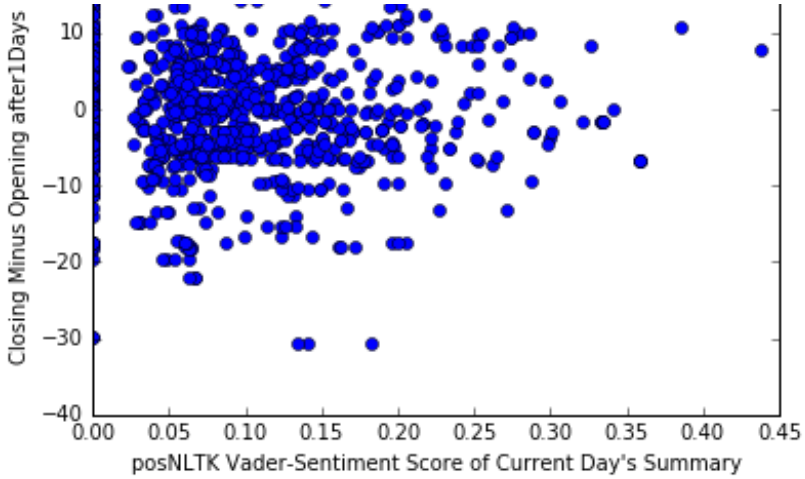
```
def sentiment_to_price_plot(stock, day_lag, pos_or_neg): #ex:
sentiment_to_price_plot("AAPL", 1, 'neg')
    stock_data = news_csv[news_csv["stock"] == stock]
    stock_price_csv = pd.read_csv(stock+"_2006-01-01_to_2017-11-01.csv")
    temp_x = []
    temp_y = []
    for index, row in stock_data.iterrows():
        ss = sid.polarity_scores(row["summary"])
        score = ss[pos_or_neg]

        day = conv_num_to_string(str(row["date"]) )

        if day in stock_price_csv["Date"].values:

            temp_x.append(score)

            row_index = stock_price_csv.index[stock_price_csv["Date"] == da
].tolist()[0]
            next_price = stock_price_csv.iloc[row_index - day_lag  ]
            #print next_price["Date"], google_price_csv.iloc[row_index]["Da
te"]
            diff = next_price["Close"] - next_price["Open"]
            temp_y.append(diff)
    print st.spearmanr(temp_x, temp_y)
    plt.plot(temp_x, temp_y, "o")
    plt.ylabel("Closing Minus Opening after" + str(day_lag) + "Days")
    plt.xlabel(pos_or_neg + "NLTK Vader-Sentiment Score of Current Day's Su
mmary")
    plt.title(stock + ",  " + str(day_lag) + " Day Lag,  " + pos_or_neg)
    plt.show()
```

In [114]:

```
sentiment_to_price_plot("GOOG", 1, "pos")
```

```
SpearmanrResult(correlation=-0.021974574026979377,
pvalue=0.45230593789238116)
```





In [17]:

```
def sentiment_scores_make_csv(stock, number_of_prices):
    stock_data = news_csv[news_csv["stock"] == stock]
    stock_price_csv = pd.read_csv("price_data/"+ stock+"_2006-01-
01_to_2017-11-01.csv")


    col = ['compound','neg','neu','pos','today price','y_price (next day)']
    col = col + [ 'today-' +str(i) + 'price' for i in
range(1,number_of_prices+1)  ]
    df = pd.DataFrame(columns=col)

    counter = 0

    for index, row in stock_data.iterrows():
        ss = sid.polarity_scores(row["summary"])
        day = conv_num_to_string(str(row["date"]) )

        if day in stock_price_csv["Date"].values:
            scores = [ss['compound'], ss['neg'], ss['neu'], ss['pos']]

            prices = []

            row_index = stock_price_csv.index[stock_price_csv["Date"] == day
].tolist()[0]
            next_price = stock_price_csv.iloc[row_index - 1  ]
            predict_closing = next_price["Close"]

            prices.append(next_price["Open"])
            prices.append(next_price["Close"])

            for j in range(1,number_of_prices+1):
                temp_price = stock_price_csv.iloc[row_index + j  ]
                prices.append(temp_price["Close"])

            total_row = scores + prices

            df.loc[counter] = total_row
            counter+=1

    name = "nltk_scores/" + stock +".csv"
    df.to_csv(name)
```

In [19]:

```
stocks = ['GOOGL', 'INTC', 'AAPL', 'CSCO', 'AMD', 'QCOM', 'NVDA', 'AMZN', 'M
SFT', 'IBM']

for stk in stocks:
    sentiment_scores_make_csv(stk, 5)
```

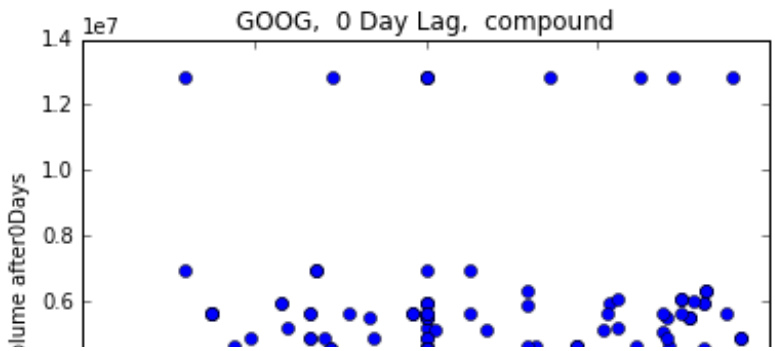
In [95]:

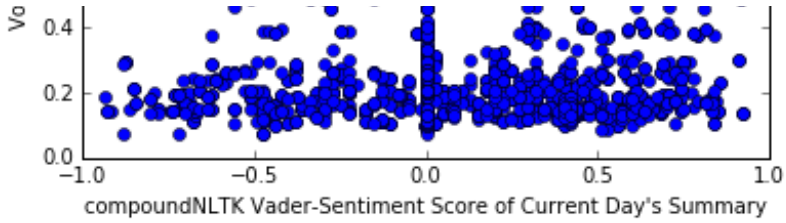
```
def sentiment_to_volume_plot(stock, day_lag, pos_or_neg): #ex:
sentiment_to_price_plot("AAPL", 1, 'neg')
    stock_data = news_csv[news_csv["stock"] == stock]
    stock_price_csv = pd.read_csv(stock+"_2006-01-01_to_2017-11-01.csv")
    temp_x = []
    temp_y = []
    for index, row in stock_data.iterrows():
        ss = sid.polarity_scores(row["summary"])
        score = ss[pos_or_neg]

        day = conv_num_to_string(str(row["date"]) )

        if day in stock_price_csv["Date"].values:

            temp_x.append(score)

            row_index = stock_price_csv.index[stock_price_csv["Date"] == da
].tolist()[0]
            next_price = stock_price_csv.iloc[row_index - day_lag  ]
            #print next_price["Date"], google_price_csv.iloc[row_index]["Da
te"]
            vol = next_price["Volume"]
            temp_y.append(vol)
    print st.spearmanr(temp_x, temp_y)
    plt.plot(temp_x, temp_y, "o")
    plt.ylabel("Volume after" + str(day_lag) + "Days")
    plt.xlabel(pos_or_neg + "NLTK Vader-Sentiment Score of Current Day's Su
mmary")
    plt.title(stock + ",  " + str(day_lag) + " Day Lag,  " + pos_or_neg)
    plt.show()
```

In [113]:

```
sentiment_to_volume_plot("GOOG", 0, 'compound')
```

```
SpearmanrResult(correlation=0.059749596923595563,
pvalue=0.040840925625703038)
```



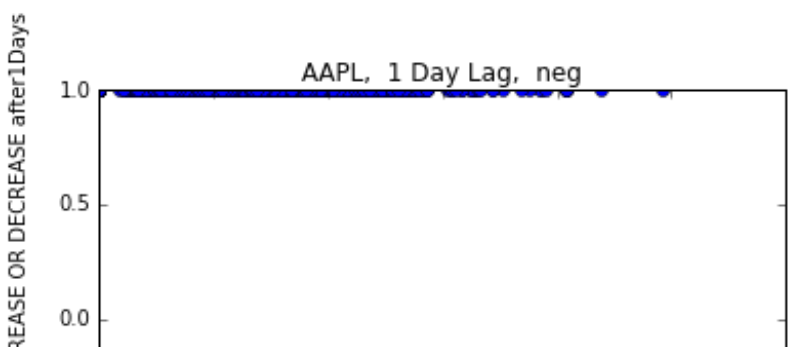


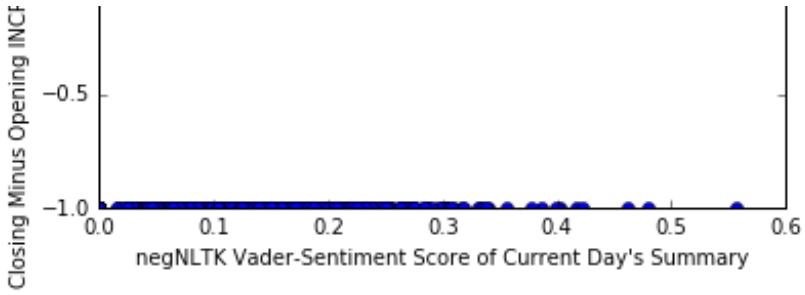
In [6]:

```
def sentiment_to_price_plot_UP_DOWN(stock, day_lag, pos_or_neg): #ex: senti
ment_to_price_plot_UP_DOWN("AAPL", 1, 'neg')
    stock_data = news_csv[news_csv["stock"] == stock]
    stock_price_csv = pd.read_csv(stock+"_2006-01-01_to_2017-11-01.csv")
    temp_x = []
    temp_y = []
    for index, row in stock_data.iterrows():
        ss = sid.polarity_scores(row["summary"])
        score = ss[pos_or_neg]

        day = conv_num_to_string(str(row["date"]) )

        if day in stock_price_csv["Date"].values:

            temp_x.append(score)

            row_index = stock_price_csv.index[stock_price_csv["Date"] == da
].tolist()[0]
            next_price = stock_price_csv.iloc[row_index - day_lag  ]
            #print next_price["Date"], google_price_csv.iloc[row_index]["Da
te"]
            diff = next_price["Close"] - next_price["Open"]
            if diff > 0:
                temp_y.append(1.0)
            else:
                temp_y.append(-1.0)

    plt.plot(temp_x, temp_y, "o")
    plt.ylabel("Closing Minus Opening INCREASE OR DECREASE after" + str(day
_lag) + "Days")
    plt.xlabel(pos_or_neg + "NLTK Vader-Sentiment Score of Current Day's Su
mmary")
    plt.title(stock + ",  " + str(day_lag) + " Day Lag,  " + pos_or_neg)
    plt.show()
```

In [7]:

```
sentiment_to_price_plot_UP_DOWN("AAPL", 1, 'neg')
```





In [40]:

```
def granger_causality(stock, max_lag, pos_or_neg):
    stock_data = news_csv[news_csv["stock"] == stock]
    stock_price_csv = pd.read_csv(stock+"_2006-01-01_to_2017-11-01.csv")
    temp_x = []
    temp_y = []
    for index, row in stock_data.iterrows():
        ss = sid.polarity_scores(row["summary"])
        score = ss[pos_or_neg]

        day = conv_num_to_string(str(row["date"]) )

        if day in stock_price_csv["Date"].values:

            temp_x.append(score)

            row_index = stock_price_csv.index[stock_price_csv["Date"] == day
].tolist()[0]
            next_price = stock_price_csv.iloc[row_index]
            #print next_price["Date"], google_price_csv.iloc[row_index]["Da
te"]
            mid_price = (next_price["Close"] + next_price["Open"])/2.0
            temp_y.append(mid_price)

#     plt.plot(temp_x)
#     plt.plot(temp_y)
#     plt.ylabel("Avg Price")
#     plt.xlabel("Time")
#     plt.title(stock + " vs " + pos_or_neg)
#     plt.show()
    return sm.tsa.stattools.grangercausalitytests([[temp_y[i], temp_x[i]] f
or i in range(len(temp_y))], maxlag = max_lag, addconst=True, verbose=True)
```

In [41]:

```
result = granger_causality("GOOG", 5, 'pos')
```

```
Granger Causality
('number of lags (no zero)', 1)
ssr based F test:         F=0.0210  , p=0.8849  , df_denom=1168, df_num=1
ssr based chi2 test:   chi2=0.0210  , p=0.8847  , df=1
likelihood ratio test: chi2=0.0210  , p=0.8847  , df=1
parameter F test:         F=0.0210  , p=0.8849  , df_denom=1168, df_num=1

Granger Causality
('number of lags (no zero)', 2)
ssr based F test:         F=0.0346  , p=0.9660  , df_denom=1165, df_num=2
ssr based chi2 test:   chi2=0.0695  , p=0.9659  , df=2
likelihood ratio test: chi2=0.0695  , p=0.9659  , df=2
```

```
parameter F test:         F=0.0346  , p=0.9660  , df_denom=1165, df_num=2

Granger Causality
('number of lags (no zero)', 3)
ssr based F test:         F=0.2290  , p=0.8762  , df_denom=1162, df_num=3
ssr based chi2 test:   chi2=0.6913  , p=0.8753  , df=3
likelihood ratio test: chi2=0.6911  , p=0.8753  , df=3
parameter F test:         F=0.2290  , p=0.8762  , df_denom=1162, df_num=3

Granger Causality
('number of lags (no zero)', 4)
ssr based F test:         F=0.3942  , p=0.8129  , df_denom=1159, df_num=4
ssr based chi2 test:   chi2=1.5892  , p=0.8107  , df=4
likelihood ratio test: chi2=1.5881  , p=0.8109  , df=4
parameter F test:         F=0.3942  , p=0.8129  , df_denom=1159, df_num=4

Granger Causality
('number of lags (no zero)', 5)
ssr based F test:         F=0.3145  , p=0.9044  , df_denom=1156, df_num=5
ssr based chi2 test:   chi2=1.5877  , p=0.9027  , df=5
likelihood ratio test: chi2=1.5866  , p=0.9029  , df=5
parameter F test:         F=0.3145  , p=0.9044  , df_denom=1156, df_num=5
```

In [49]:

```
result
```

Out[49]:

```
{1: ({'lrtest': (0.021018253823967825, 0.8847292401249347, 1),
   'params_ftest': (0.020964595036337345, 0.88490042830568649, 1168.0, 1),
   'ssr_chi2test': (0.021018442453445682, 0.88472872648798118, 1),
   'ssr_ftest': (0.020964595034692193, 0.88490042830992832, 1168.0, 1)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
48272d0>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
4827c10>,
   array([[ 0.,  1.,  0.]])]),
 2: ({'lrtest': (0.06948480469873175, 0.96585418598410699, 2),
   'params_ftest': (0.034594957809702974, 0.96599759874417668, 1165.0, 2),
   'ssr_chi2test': (0.069486868046849642, 0.96585318953791255, 2),
   'ssr_ftest': (0.034594957809649503, 0.96599759874430158, 1165.0, 2)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
492aed0>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
492a810>,
   array([[ 0.,  0.,  1.,  0.,  0.],
          [ 0.,  0.,  0.,  1.,  0.]])]),
 3: ({'lrtest': (0.6910776299328063, 0.87530054736263196, 3),
   'params_ftest': (0.22904750983883956, 0.87620224217496578, 1162.0, 3),
   'ssr_chi2test': (0.6912819423438038, 0.87525258367736658, 3),
   'ssr_ftest': (0.22904750983846595, 0.87620224217523845, 1162.0, 3)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
492a490>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
492a5d0>,
   array([[ 0.,  0.,  0.,  1.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  1.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  1.,  0.]])]),
 4: ({'lrtest': (1.5881026373263012, 0.81092885826865435, 4),
   'params_ftest': (0.39423434120240486, 0.81287965504549664, 1159.0, 4),
```

```
   'ssr_chi2test': (1.5891827800680756, 0.81073500282313649, 4),
   'ssr_ftest': (0.39423434120267542, 0.8128796550452887, 1159.0, 4)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
492af90>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
490e750>,
   array([[ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.]])]),
 5: ({'lrtest': (1.5865823297117458, 0.90286715845947607, 5),
   'params_ftest': (0.31453924532518884, 0.90441548464785937, 1156.0, 5),
   'ssr_chi2test': (1.5876613291292585, 0.90273741475789537, 5),
   'ssr_ftest': (0.31453924532535099, 0.90441548464776711, 1156.0, 5)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
490e0d0>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
490e510>,
   array([[ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.]])])}
```

In [52]:

```
sm.tsa.stattools.grangercausalitytests(x = np.array([[1,2],[2,3],[3,4], [4,5
], [5,6], [7,8], [8,9]  ]), maxlag = 1, addconst=True, verbose=True)
```

```
Granger Causality
('number of lags (no zero)', 1)
ssr based F test:         F=0.0000  , p=1.0000  , df_denom=4, df_num=1
ssr based chi2 test:   chi2=0.0000  , p=1.0000  , df=1
likelihood ratio test: chi2=0.0000  , p=1.0000  , df=1
parameter F test:         F=45.4683 , p=0.0025  , df_denom=4, df_num=1
```

Out[52]:

```
{1: ({'lrtest': (3.5527136788005009e-15, 0.99999995244237416, 1),
   'params_ftest': (45.46826758147516, 0.0025211364311817572, 4.0, 1),
   'ssr_chi2test': (1.7595987560096852e-15, 0.99999996653068035, 1),
   'ssr_ftest': (1.1730658373397901e-15, 0.99999997764825821, 4.0, 1)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
491abd0>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d0
491aa10>,
   array([[ 0.,  1.,  0.]])])}
```