In [ ]:
```
import datetime
import pandas as pd
import pandas_datareader.data as web
```

In [ ]:
```
start = datetime.datetime(2004, 8, 19) # or start = '1/1/2016'
end = datetime.date.today()
df = web.DataReader('GOOG', 'yahoo', start, end)
print (df.head())  # print first rows of the prices data
```

In [1]:
```
import time
import math
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers.recurrent import LSTM
import numpy as np
import pandas as pd
import tensorflow as tf
import sklearn.preprocessing as prep
from keras import backend
```

```
Using TensorFlow backend.
```

In [2]:
```
import os
s=os.getcwd()
s
```

Out[2]:
```
'/Users/Yuffie/USA/SCU/COEN281DataMining/TermProject'
```

In [3]:
```
df = pd.read_csv('/Users/Yuffie/USA/SCU/COEN281DataMining/TermProject/data/GOOG.csv')
df.head()
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	49.676899	51.693783	47.669952	49.845802	49.845802	44994500
1	2004-08-20	50.178635	54.187561	49.925285	53.805050	53.805050	23005800
2	2004-08-23	55.017166	56.373344	54.172661	54.346527	54.346527	18393200
3	2004-08-24	55.260582	55.439419	51.450363	52.096165	52.096165	15361800
4	2004-08-25	52.140873	53.651051	51.604362	52.657513	52.657513	9257400

In [4]:
```
# Data preparation
col_list = df.columns.tolist()
col_list
```

Out[4]: ['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']

In [5]:
```
col_list.remove('Close')
col_list.append('Close')
col_list.remove('Date')
col_list.remove('Volume')
col_list
```

Out[5]: ['Open', 'High', 'Low', 'Adj Close', 'Close']

In [6]:
```
df = df[col_list]
df.head()
```

Out[6]:

	Open	High	Low	Adj Close	Close
0	49.676899	51.693783	47.669952	49.845802	49.845802
1	50.178635	54.187561	49.925285	53.805050	53.805050
2	55.017166	56.373344	54.172661	54.346527	54.346527
3	55.260582	55.439419	51.450363	52.096165	52.096165
4	52.140873	53.651051	51.604362	52.657513	52.657513

In [7]:
```
# Save data
df.to_csv('GOOG-adjust.csv', index=False)
validate_df = pd.read_csv('GOOG-adjust.csv')
validate_df.head()
```

Out[7]:

	Open	High	Low	Adj Close	Close
0	49.676899	51.693783	47.669952	49.845802	49.845802
1	50.178635	54.187561	49.925285	53.805050	53.805050
2	55.017166	56.373344	54.172661	54.346527	54.346527
3	55.260582	55.439419	51.450363	52.096165	52.096165
4	52.140873	53.651051	51.604362	52.657513	52.657513

In [8]:
```
# Standardization the dataset
def standard_scaler(X_train, X_test):
    train_samples, train_nx, train_ny = X_train.shape
    test_samples, test_nx, test_ny = X_test.shape

    X_train = X_train.reshape((train_samples, train_nx * train_ny))
    X_test = X_test.reshape((test_samples, test_nx * test_ny))

    preprocessor = prep.StandardScaler().fit(X_train)
    X_train = preprocessor.transform(X_train)
    X_test = preprocessor.transform(X_test)

    X_train = X_train.reshape((train_samples, train_nx, train_ny))
    X_test = X_test.reshape((test_samples, test_nx, test_ny))

    return X_train, X_test
```

In [9]:
```
# Split the data to X_train, y_train, X_test, y_test
def preprocess_data(stock, seq_len):
    amount_of_features = len(stock.columns)
    data = stock.as_matrix()

    sequence_length = seq_len + 1
    result = []
    for index in range(len(data) - sequence_length):
        result.append(data[index : index + sequence_length])

    result = np.array(result)
    row = round(0.9 * result.shape[0])
    train = result[: int(row), :]
    y_test_org = result[int(row) :, -1][ : ,-1]

    train, result = standard_scaler(train, result)

    X_train = train[:, : -1]
    y_train = train[:, -1][: ,-1]
    X_test = result[int(row) :, : -1]
    y_test = result[int(row) :, -1][ : ,-1]

    X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], amount_of_features))
    X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], amount_of_features))

    return [X_train, y_train, X_test, y_test, y_test_org]
```

In [10]:
```
# Build LSTM Neural Network
# LSTM --> Dropout --> LSTM --> Dropout --> Fully-Conneted(Dense)
def build_model(layers):
    model = Sequential()

    # By setting return_sequences to True we are able to stack another LSTM layer
    model.add(LSTM(
        return_sequences=True,
        input_shape=(None, 5), units=20))
    model.add(Dropout(0.4))

    model.add(LSTM(
        layers[2],
        return_sequences=False))
    model.add(Dropout(0.3))

    model.add(Dense(
        units=1))
    model.add(Activation("linear"))

    start = time.time()
    model.compile(loss="mse", optimizer="rmsprop", metrics=['accuracy'])
    print("Compilation Time : ", time.time() - start)
    return model
```

In [11]:
```
window = 20
X_train, y_train, X_test, y_test, y_test_org = preprocess_data(df[:: -1], window)
print("X_train", X_train.shape)
print("y_train", y_train.shape)
print("X_test", X_test.shape)
print("y_test", y_test.shape)
```

```
X_train (2993, 20, 5)
y_train (2993,)
X_test (333, 20, 5)
y_test (333,)
```

In [12]:
```
model = build_model([X_train.shape[2], window, 100, 1])
```

```
Compilation Time :  0.027528762817382812
```

In [13]:
```
# Training the model
model.fit(
    X_train,
    y_train,
    batch_size=768,
    epochs=300,
    validation_split=0.1,
    verbose=0)
```

Out[13]:
```
<keras.callbacks.History at 0x11a338f60>
```

In [14]:
```
trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))
print(model.metrics_names)
testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))
```

```
Train Score: 0.01 MSE (0.10 RMSE)
['loss', 'acc']
Test Score: 0.03 MSE (0.18 RMSE)
Test Accuracy:  0.0
```

In [15]:
```
#Visualize the Prediction
diff = []
ratio = []
pred = model.predict(X_test)
for u in range(len(y_test)):
    pr = pred[u][0]
    ratio.append((y_test[u] / pr) - 1)
    diff.append(abs(y_test[u] - pr))
print('error_ratio', ratio)
#print('error_abs', diff)
#print(pred)
```

```
error_ratio [0.1133188270722516, 0.12113117333001178, 0.12434312271378367, 0.11442692494278739, 0.1257696360
1427349, 0.11864027114397868, 0.11005930226036575, 0.075574684509538859, 0.084678743835904458, 0.10894661668
503991, 0.11055392153412047, 0.056963478914290544, 0.046109114754503411, 0.066633294650770392, 0.08816201648
1074135, 0.10856129348853449, 0.13054266722684593, 0.1165475983346318, 0.12465535042704268, 0.13225768293636
753, 0.11455260024952674, 0.12563889971341391, 0.11823949618952434, 0.14284994499788484, 0.1083994796314171
6, 0.090535262817867723, 0.10001035004008019, 0.10948303129434866, 0.12223594338050647, 0.11856326514441684,
0.13144598700601251, 0.15916624908582722, 0.16153487170182634, 0.14432027095464384, 0.15431753890243072, 0.1
3962986656818854, 0.15097689495951205, 0.22747057581207875, 0.19826187375467286, 0.19251029015459498, 0.1719
0272651737093, 0.17728446177315971, 0.16145640026881947, 0.14286161875122194, 0.12372685151387786, 0.1090369
9823435731, 0.10164919702920261, 0.10143304131299202, 0.1055631708226239, 0.10462790667290789, 0.08828725015
6050101, 0.093981031494910106, 0.10969986917575847, 0.11725084849272149, 0.099319146574329054, 0.09850793309
2572886, 0.097184536193824833, 0.10694249089782559, 0.1066054627436015, 0.115194293414017l, 0.12312947177092
926, 0.1287291464428193, 0.12064625873545287, 0.1172149522739927, 0.097158244113885628, 0.10130804463347087,
0.12438471434538401, 0.13071988780762633, 0.1288259142970849, 0.14146556753485195, 0.1338802128607843 3, 0.13
387965311358263, 0.13014609254819542, 0.12385212671328905, 0.11858105687081233, 0.126531175852856, 0.1262155
4365289134, 0.12371377577806708, 0.12744016752832832, 0.13612256441420767, 0.12081260997149834, 0.1184098808
7697202, 0.10636171346598444, 0.10482117402303426, 0.10867015963991822, 0.097927441786243907, 0.110706718453
77178, 0.10802162770624291, 0.096621968191945573, 0.098556446825680988, 0.098097506002420287, 0.089152445514
823375, 0.092570489372345577, 0.091348510223351553, 0.10949157557155154, 0.1184579889829962, 0.1062985084952
8277, 0.099257386819508797, 0.10191468584838681, 0.10358741077375044, 0.091388412273360586, 0.06975320710360
8217, 0.078062620506814895, 0.087004426286523051, 0.11220802164313737, 0.10970732199041655, 0.11011249805921
408, 0.1140910882003729 5, 0.12804106128304471, 0.12257395826464101, 0.11426562179350586, 0.114143106866833 0
4, 0.12135005578782243, 0.11133121797146228, 0.11962064619428925, 0.11247368569271643, 0.11463196310673252,
0.095071312878590719, 0.091852112004521702, 0.10795228595498685, 0.12432894897040248, 0.12398650268416311,
0.12482022328461828, 0.12473578024807841, 0.13534651091571992, 0.13761103342262726, 0.13891392289899041, 0.1
2743072852657811, 0.11518325438874988, 0.11399447046671352, 0.10560213336956137, 0.11955340204955145, 0.0921
51173878707748, 0.098113292712962 43, 0.12176631295892615, 0.10636598298099842, 0.10608306096046705, 0.128386
46091786154, 0.14998534141424091, 0.1595527465994302, 0.15022396079456324, 0.15380239796763018, 0.1493588705
2008456, 0.17173765892932913, 0.16969945467220682, 0.16233634799055063, 0.16728157159875789, 0.1645164263405
2344, 0.161463390679355, 0.15651297800891562, 0.14646766677179324, 0.14908917616968642, 0.14882124062175772,
0.141604917211132, 0.14077514002591407, 0.13561375527111785, 0.13824921264963907, 0.14406473072979975, 0.146
40907691791183, 0.14518105075958698, 0.14231959175079867, 0.14227388107361882, 0.13169303117352071, 0.145273
1696212155, 0.16599939161852517, 0.17452655768404246, 0.18236316815038522, 0.1848642917025547, 0.18234112076
835918, 0.16485125740452622, 0.15796079851647171, 0.15336914900261567, 0.15287701395254527, 0.15359318192063
554, 0.14922872971983403, 0.15632133667237569, 0.15600327851635876, 0.16024132651421974, 0.1677236578895455
7, 0.16436860687232357, 0.16191575602820474, 0.16124537200350941, 0.15620285434710746, 0.1588638294607494,
0.15831727588603961, 0.15802949814080347, 0.15313531258879554, 0.15407437499206034, 0.15491660350933611, 0.1
6093121865139937, 0.15474449856398031, 0.16040381127494996, 0.15462214651782724, 0.150157400742875, 0.14760
607895037192, 0.1411505961216899, 0.13574809334534166, 0.14239398541615511, 0.14176706864155109, 0.146063962
49793785, 0.14520575721869511, 0.14236899618070153, 0.14677429990448676, 0.14194860400972775, 0.133564468269
46389, 0.13928053404385432, 0.12900244737249889, 0.13082495268985239, 0.13189082549965403, 0.139432220535972
15, 0.14477815653848025, 0.15529025705344135, 0.15364931322913544, 0.14624821931120158, 0.13338507992084425,
```

```
0.13880652627899659, 0.12519887675323282, 0.11542357612988541, 0.12717126134501089, 0.15560366732458242, 0.1
4950994063135004, 0.15825699766686152, 0.16092780466168932, 0.15678487186743872, 0.17672980410378036, 0.1674
8271585781493, 0.15128635951276159, 0.13992194601888985, 0.13402770535358854, 0.12359198496382739, 0.1328665
5763769328, 0.14314819279481417, 0.14385677537929409, 0.1472155624118261, 0.14412771746223063, 0.14600565836
431945, 0.15515498161762675, 0.14552306353655498, 0.14323712477653516, 0.1287130238205938, 0.147757014717187
16, 0.13993042992392168, 0.14884767789061204, 0.14896264473076792, 0.14997815331664865, 0.15654611497504956,
0.15826037608391497, 0.16129701959532228, 0.15731200483823082, 0.16468057962750615, 0.16932265570901994, 0.1
6121146093794803, 0.16118715056978838, 0.17426556488530776, 0.16988807834810271, 0.16437299734156263, 0.1685
456555345477, 0.16420629049699786, 0.15415255493304492, 0.14625562883581655, 0.14824412991573288, 0.14783147
407647657, 0.14487152841661688, 0.14720189798935057, 0.15077684008444869, 0.15922301510983949, 0.17163535703
985344, 0.17448969680822612, 0.16481663483320075, 0.16620699535604455, 0.15631215185561831, 0.15579418302141
956, 0.13425194250306927, 0.14070871267349783, 0.14097375400492251, 0.1694651206793869, 0.16774144292360349,
0.15962631227039359, 0.16405879742671026, 0.13633217644446538, 0.12478173982516316, 0.12187389479433608, 0.1
2337844400843889, 0.13647372502196609, 0.13451072261689601, 0.14951437865673878, 0.15769677022123463, 0.1474
9711406343735, 0.17411876538044169, 0.21378779971440998, 0.2245964789007675, 0.20409076825660999, 0.19522722
592484176, 0.19897990728575565, 0.19832901769227562, 0.19633026924419261, 0.19885816535971834, 0.19911727333
859308, 0.19143277624491772, 0.18650214048406299, 0.18718944083107969, 0.18296792730318256, 0.18697594490492
264, 0.18956682048592599, 0.1929037796635511, 0.18842957203659116, 0.19392617715564708, 0.20673912009707363,
0.2016336967655632, 0.19749690021781729, 0.19965503247535432, 0.19879625021665781, 0.19465105537454064, 0.19
661673568056282, 0.20144793500142799, 0.20349226538433562, 0.20285479695905617, 0.20798332458387869, 0.20984
301497551594, 0.21301765046941279, 0.21094281539512894, 0.21018546408171712, 0.21108186815549135, 0.20709266
282994254, 0.20797705657082366, 0.20349848680961746, 0.20338556181727641, 0.19615581794240855, 0.19322983427
300011, 0.19674655598703406, 0.19898090268026247, 0.19174611833834221, 0.19391141030298531]
```

In [16]:
```
# Scale the representation back
y_test = y_test_org
pred_org = []
for u in range(len(y_test)):
    pred_org.append(y_test[u]/(ratio[u] + 1))
    print(ratio[u])
#print(pred_org)
#print(y_test)
```

```
0.113318827072
0.12113117333
0.124343122714
0.114426924943
0.125769636014
0.118640271144
0.11005930226
0.0755746845095
0.0846787438359
0.108946616685
0.110553921534
0.0569634789143
0.0461091147545
0.0666332946508
0.0881620164811
0.108561293489
0.130542667227
0.116547598335
0.124655350427
0.132257682936
0.11455260025
0.125638899713
0.11823949619
0.142849944998
0.108399479631
0.0905352628179
0.10001035004
0.109483031294
0.122235943381
0.118563265144
0.131445987006
0.159166249086
0.161534871702
0.144320270955
0.154317538902
0.139629866568
0.15097689496
0.227470575812
0.198261873755
0.192510290155
0.171902726517
0.177284461773
```

```
0.161456400269
0.142861618751
0.123726851514
0.109036998234
0.101649197029
0.101433041313
0.105563170823
0.104627906673
0.0882872501561
0.0939810314949
0.109699869176
0.117250848493
0.0993191465743
0.0985079330926
0.0971845361938
0.106942490898
0.106605462744
0.115194293414
0.123129471771
0.128729146443
0.120646258735
0.117214952274
0.0971582441139
0.101308044633
0.124384714345
0.130719887808
0.128825914297
0.141465567535
0.133880212861
0.133879653114
0.130146092548
0.123852126713
0.118581056871
0.126531175853
0.126215543653
0.123713775778
0.127440167528
0.136122564414
0.120812609971
0.118409880877
0.106361713466
0.104821174023
```

```
0.10867015964
0.0979274417862
0.110706718454
0.108021627706
0.0966219681919
0.0985564468257
0.0980975060024
0.0891524455148
0.0925704893723
0.0913485102234
0.109491575572
0.118457988983
0.106298508495
0.0992573868195
0.101914685848
0.103587410774
0.0913884122734
0.0697532071036
0.0780626205068
0.0870044262865
0.112208021643
0.10970732199
0.110112498059
0.1140910882
0.128041061283
0.122573958265
0.114265621794
0.114143106867
0.121350055788
0.111331217971
0.119620646194
0.112473685693
0.114631963107
0.0950713128786
0.0918521120045
0.107952285955
0.12432894897
0.123986502684
0.124820223285
0.124735780248
0.135346510916
0.137611033423
```

```
0.138913922899
0.127430728527
0.115183254389
0.113994470467
0.10560213337
0.11955340205
0.0921511738787
0.098113292713
0.121766312959
0.106365982981
0.10608306096
0.128386460918
0.149985341414
0.159552746599
0.150223960795
0.153802397968
0.14935887052
0.171737658929
0.169699454672
0.162336347991
0.167281571599
0.164516426341
0.161463390679
0.156512978009
0.146467666772
0.14908917617
0.148821240622
0.141604917211
0.140775140026
0.135613755271
0.13824921265
0.14406473073
0.146409076918
0.14518105076
0.142319591751
0.142273881074
0.131693031174
0.145273169621
0.165999391619
0.174526557684
0.18236316815
0.184864291703
```

```
0.182341120768
0.164851257405
0.157960798516
0.153369149003
0.152877013953
0.153593181921
0.14922872972
0.156321336672
0.156003278516
0.160241326514
0.16772365789
0.164368606872
0.161915756028
0.161245372004
0.156202854347
0.158863829461
0.158317275886
0.158029498141
0.153135312589
0.154074374992
0.154916603509
0.160931218651
0.154744498564
0.160403811275
0.154622146518
0.150157740074
0.14760607895
0.141150596122
0.135748093345
0.142393985416
0.141767068642
0.146063962498
0.145205757219
0.142368996181
0.146774299904
0.14194860401
0.133564468269
0.139280534044
0.129002447372
0.13082495269
0.1318908255
0.139432220536
```

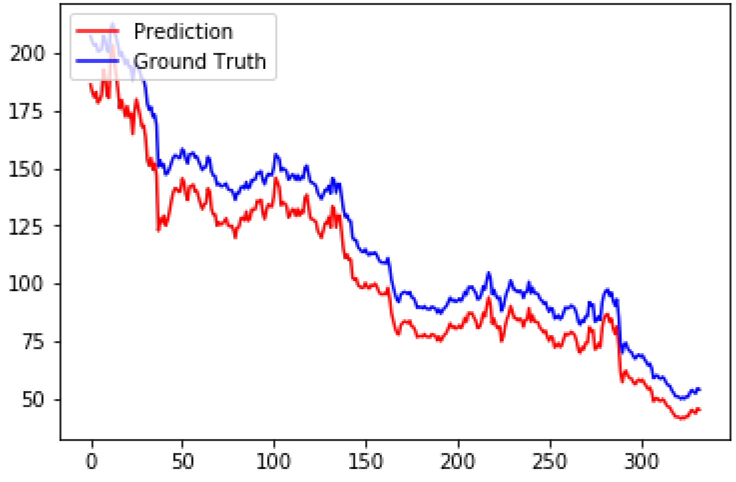
```
0.144778156538
0.155290257053
0.153649313229
0.146248219311
0.133385079921
0.138806526279
0.125198876753
0.11542357613
0.127171261345
0.155603667325
0.149509940631
0.158256997667
0.160927804662
0.156784871867
0.176729804104
0.167482715858
0.151286359513
0.139921946019
0.134027705354
0.123591984964
0.132866557638
0.143148192795
0.143856775379
0.147215562412
0.144127717462
0.146005658364
0.155154981618
0.145523063537
0.143237124777
0.128713023821
0.147757014717
0.139930429924
0.148847677891
0.148962644731
0.149978153317
0.156546114975
0.158260376084
0.161297019595
0.157312004838
0.164680579628
0.169322655709
0.161211460938
```

```
0.16118715057
0.174265564885
0.169888078348
0.164372997342
0.168545655535
0.164206290497
0.154152554933
0.146255628836
0.148244129916
0.147831474076
0.144871528417
0.147201897989
0.150776840084
0.15922301511
0.17163535704
0.174489696808
0.164816634833
0.166206995356
0.156312151856
0.155794183021
0.134251942503
0.140708712673
0.140973754005
0.169465120679
0.167741442924
0.15962631227
0.164058797427
0.136332176444
0.124781739825
0.121873894794
0.123378444008
0.136473725022
0.134510722617
0.149514378657
0.157696770221
0.147497114063
0.17411876538
0.213787799714
0.224596478901
0.204090768257
0.195227225925
0.198979907286
```

```
0.198329017692
0.196330269244
0.19885816536
0.199117273339
0.191432776245
0.186502140484
0.187189440831
0.182967927303
0.186975944905
0.189566820486
0.192903779664
0.188429572037
0.193926177156
0.206739120097
0.201633696766
0.197496900218
0.199655032475
0.198796250217
0.194651055375
0.196616735681
0.201447935001
0.203492265384
0.202854796959
0.207983324584
0.209843014976
0.213017650469
0.210942815395
0.210185464082
0.211081868155
0.20709266283
0.207977056571
0.20349848681
0.203385561817
0.196155817942
0.193229834273
0.196746555987
0.19898090268
0.191746118338
0.193911410303
```

In [17]:
```
import matplotlib.pyplot as plt2

plt2.plot(pred_org, color='red', label='Prediction')
plt2.plot(y_test, color='blue', label='Ground Truth')
plt2.legend(loc='upper left')
plt2.show()
```



In [ ]: