NewzTrader: Autonomous Trading Agent Implementation Using Natural Language Processing Of News Headlines

William Lyon

November 18, 2012

Abstract

Natural Language Processing techniques are used to examine financial news headlines and generate predictions of stock price movements. News headlines from the Wall Street Journal from January 1, 2009 to November 1, 2012 are collected and paired with daily S&P 500 index returns. This information is used to train a Naive Bayes classifier and generate BUY/SELL trading signals. This trading strategy is then backtested using Wall Street Journal news headlines as a predictor for movement in the price of the S&P 500 index.

1 Introduction

In finance, the Efficient Market Hypothesis states that all publicly available information is reflected in financial market prices. As new information becomes available, prices adjust to take this new information into account.

1.1 Motivation

This tool could be used as a component of an autonomous trading agent that will make BUY/SELL decisions for trading financial instruments.

Goals: 1)

1.2 Literature Review

2 Examining The Data

Google Finance provides access to historical stock quotes as well as links to financial news stories pertaining to specific stocks. This data provides the basis for this paper.

2.1 Historical Stock Market Quotes

2.2 News Headlines

A total of 101,618 news headlines over 1416 days were collected with an average of 71 headlines per day. After aligning with financial data: UP: 14651 DOWN: 13718 NONE: 57405 TOTOAL: 85775

Prior probs: UP: 0.1708 DOWN: 0.1599 NONE: 0.6693

3 Classification Methodology

These tuples of [return, listOfDailyNews] are used as the training data for a Naive Bayes classifier.

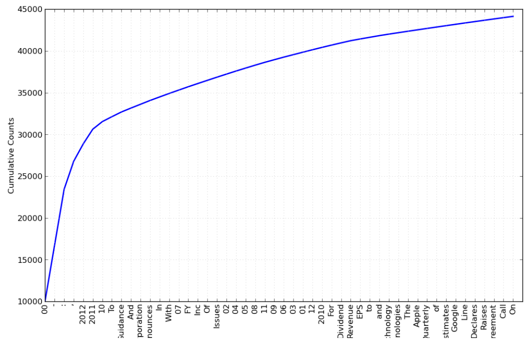

Figure 1: Cumulative frequency plot

3.1 Classes

3.2 Naive Bayes

3.3 Maximum Entropy

4 Implementation

4.1 Dependencies

4.2 Data Collection & Munging

4.3 Feature Extraction

4.3.1 Bag of words

4.3.2 Filtering stopwords

4.3.3 Include significant bigrams

4.4 Naive Bayes Classifier

4.5 Maximum Entropy Classifier

4.6 Backtesting

5 Evaluation

5.1 Accuracy

5.2 Most informative features

5.3 Precision

5.4 Recall

6 Trading Model

6.1 Trading Signals

6.2 Backtesting

9 Source Code Listings

Listing 1: newsCredScraper.py: scrapes WSJ news headlines

```

from lxml import etree
from datetime import datetime
from datetime import date
from datetime import timedelta
#import datetime
from dateutil.parser import parse
#import pandas as pd
import pickle
#from pandas.io.data import DataReader
#from pandas.io.data import DataReader
#from pandas.io.data import DataReader

#import pandas as pd
#from datetime import datetime

#iimport pandas as pd
#
#from pandas.io.data import DataReader

def daterange(start_date, end_date):
    for n in range(int ((end_date - start_date).days)):
        yield start_date + timedelta(n)

def only_alphanum(s):
    #s = unicode(s, "utf-8")
    return '␣'.join(c for c in s.split() if c.isalnum())
def only_alpha(s):
    return '␣'.join(c for c in s.split() if c.isalpha())
def removeNonAscii(s): return "".join(i for i in s if ord(i)<128)

#sp500 = DataReader("SPY", "yahoo", datetime(2009, 1, 1))
news={}
#descs = {}
start_date=date(2011, 7, 27)
end_date = date(2012, 11, 9)
news[date(2009, 11, 9)] = []
news[date(2009, 11, 10)] = []
for single_date in daterange(start_date, end_date):
    news[single_date]=[]
for single_date in daterange(start_date, end_date):
    #print "Starting date: " + str(single_date)
    year = single_date.year
    month = single_date.month
    day = single_date.day
    path = "http://api.newscred.com/articles?access_key=c4bcc3f7c9bf9ec159f51da0a86ca658&sources=104afa30d811d37a5582a39e1662a311&pagesize=99&from_date=%d-%d-%d&to_date=%d-%d-%d␣23:59:59" % (year, month, day, year, month, day)
    while True:
        try:
            root = etree.parse(path)
            break
        except etree.XMLSyntaxError:
            pass
    myRoot = root.getroot()

    #news[date(year, month, day)]=[]
    #descs[date(year, month, day)]=[]
    for element in myRoot.iter("article"):
        #for item in element.iter("description"):
         # desc = item.text
        for item in element.iter("title"):
            title = item.text
        for item in element.iter("created_at"):
            pubDate = parse(item.text)

        news[pubDate.date()].append(only_alphanum(removeNonAscii(title)))
#import pickle
    output = open('newsDict2.pkl', 'wb')
    pickle.dump(news, output)
    output.close()
```

Listing 2: dataGetter.py: downloads historical S&P 500 index price data and joins with WSJ news headlines / identifies news headline classifications and saves in format for corpus reader

```
# dataGetter.py
# William Lyon
# AI Grad Project
# NewzTrader
# dataGetter.py
# 1) Loads pickled news dict
# 2) Downloads historical stock price data
# 3) Joins news dict and stock data in a pandas dataframe
# 4) Set UP/DOWN/NONE classifications for every trading day
# 5) Save news headlines in .CSV files for corpus reader

import pickle
from datetime import datetime
from datetime import timedelta
from datetime import date

def daterange(start_date, end_date):
    for n in range(int ((end_date - start_date).days)):
        yield start_date + timedelta(n)

fkl_file = open('combinedNewsDictFull.pkl', 'rb')
news = pickle.load(fkl_file)
fkl_file.close()
start_date=date(2009, 1, 1)
end_date = date(2012, 11, 17)

import pandas as pd
#
from pandas.io.data import DataReader

def only_alphanum(s):
    #s = unicode(s, "utf-8")
    return ' '.join(c for c in s.split() if c.isalnum())
def only_alpha(s):
    return ' '.join(c for c in s.split() if c.isalpha())
def removeNonAscii(s): return "".join(i for i in s if ord(i)<128)

sp500 = DataReader("^GSPC", "yahoo", datetime(2009, 1, 1))
newsframe = pd.Series(news, name='News')
#descframe = pd.Series(descs, name='Desc')
frameWithNews = sp500.join(pd.DataFrame(newsframe))
#sp500Frame = frameWithNews.join(pd.DataFrame(descframe))
newsframe = pd.Series(news, name='News')
#descframe = pd.Series(descs, name='Desc')
frameWithNews = sp500.join(pd.DataFrame(newsframe))
```

```

newsReturns = frameWithNews['Adj_Close'].pct_change()
newsReturns.name='Returns'
returnsFrame = frameWithNews.join(pd.DataFrame(newsReturns))

returnsFrame['UP'] = returnsFrame.Returns > 0.01
returnsFrame['DOWN'] = returnsFrame.Returns < -0.01
returnsFrame['NONE'] = (returnsFrame['UP']==False) & (returnsFrame['DOWN']==False)

droppedFrame = returnsFrame

newsUP_frame = droppedFrame[droppedFrame['UP']==True]


newsDOWN_frame = droppedFrame[droppedFrame['DOWN']==True]
newsNONE_frame = droppedFrame[droppedFrame['NONE']==True]

newsNONE_frame = newsNONE_frame.dropna()
newsUP_frame = newsUP_frame.dropna()
newsDOWN_frame = newsDOWN_frame.dropna()

# DO THIS FOR NONE, UP, and DOWN
i = 1
for row in newsNONE_frame.iterrows():
    i+=1
    if len(row[1].ix['News'])>0:
        for line in row[1].ix['News']:
            i+=1
            writeFile = open('%d_news_NONE.csv' % i, 'w')
            writeFile.write(line+'\n')

# DO THIS FOR NONE, UP, and DOWN
i = 1
for row in newsUP_frame.iterrows():
    i+=1
    if len(row[1].ix['News'])>0:
        for line in row[1].ix['News']:
            i+=1
            writeFile = open('%d_news_UP.csv' % i, 'w')
            writeFile.write(line+'\n')

# DO THIS FOR NONE, UP, and DOWN
i = 1
for row in newsDOWN_frame.iterrows():
    i+=1
    if len(row[1].ix['News'])>0:
        for line in row[1].ix['News']:
            i+=1
            writeFile = open('%d_news_DOWN.csv' % i, 'w')
            writeFile.write(line+'\n')
```

Listing 3: nbTrainer.py: loads news corpus / trains Naive Bayes classifier

```
# nbTrainer.py
# William Lyon
# AI Grad Project
# NewzTrader
# nbTrainer.py
# 1) Load NLTK corpus reader for naive bayes classifier
# 2) Train NB classifier with random 90% of corpus features
# 3) Test NB classifier with remaining 10% of corpus features and report
# accuracy
# TODO: recall, precision reports; improve classifier (only strong words?)

# NLTK - train nb_classifier


import random
import nltk as nltk
#nltk.download()
from nltk.corpus import stopwords
import os, os.path
path = os.path.expanduser('~/nltk_data')
if not os.path.exists(path):
    os.mkdir(path)
os.path.exists(path)
import nltk.data
path in nltk.data.path
from nltk.corpus.reader import CategorizedPlaintextCorpusReader
reader = CategorizedPlaintextCorpusReader('.', r'.*_news_.*\.csv', cat_pattern=r'.*_news_(\w+)\.csv')
reader.categories()

def bag_of_words(words):
    return dict([(word, True) for word in words if word[0].isalpha()])
import collections
def bag_of_words_not_in_set(words, badwords):
    return bag_of_words(set(words)-set(badwords))

def bag_of_non_stopwords(words, stopfile='english'):
    badwords = stopwords.words(stopfile)
    return bag_of_words_not_in_set(words, badwords)

from nltk.metrics import BigramAssocMeasures
from nltk.collocations import BigramCollocationFinder

def bag_of_bigrams_words(words, score_fn=BigramAssocMeasures.chi_sq, n=2000):
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(score_fn, n)
    dictOfBigrams = bag_of_words(bigrams)
    dictOfBigrams.update(bag_of_non_stopwords(words))
    return dictOfBigrams

def label_feats_from_corpus(corp, feature_detector=bag_of_bigrams_words):
    label_feats = collections.defaultdict(list)
    for label in corp.categories():
        for fileid in corp.fileids(categories=[label]):
            feats = feature_detector(corp.words(fileids=[fileid]))
            label_feats[label].append(feats)
    return label_feats

def split_label_feats(lfeats, split=0.90):
    train_feats = []
    test_feats = []
    for label, feats in lfeats.iteritems():
        random.shuffle(feats, random.random)
        cutoff = int(len(feats) * split)
        train_feats.extend([(feat, label) for feat in feats[:cutoff]])
        test_feats.extend([(feat, label) for feat in feats[cutoff:]])
    return train_feats, test_feats




```

```
reader.categories()

lfeats = label_feats_from_corpus(reader)
lfeats.keys()
train_feats, test_feats = split_label_feats(lfeats)
len(train_feats)
len(test_feats)

from nltk.classify import NaiveBayesClassifier
nb_classifier = NaiveBayesClassifier.train(train_feats)
nb_classifier.labels()

from nltk.classify.util import accuracy
accuracy(nb_classifier, test_feats)
```

Listing 4: backTest.py: simulates trading using trading signals generated from WSJ news headlines using Naive Bayes classifier

Listing 5: NewzTrader.py