
CAPSTONE PROJECT

Master of Data Science Program – Fall 2016

DECEMBER 14

BY

Wensi Hu

Table of Contents

1. Abstract.....	3
2. Motivation	3
3. Data Acquisition	4
4. Data Pre-Processing.....	5
4.1 Natural Language Processing	5
4.2 Name Entity	6
4.3 Special Focus on Sentiment Variable	7
5. Data Exploration.....	7
5.1 Divide data set into Training/Testing data	7
5.2 Simple glance	9
6. Predictive Model Training.....	9
6.1 LSTM Model building.....	9
6.2 Model Training and Validation.....	10
7. Conclusion	11
7.1 Project summary	11
7.2 Future adjustment.....	11
8. Reference	12
9. Appendix	12

1. Abstract

The objective of this project is to create a Sentiment Predictive Analysis System based specially on news text data that includes data streaming, data pre-processing, feature selection and sentiment predictive model training.

This project includes four parts:

1. Data Acquisition: In the beginning of this project, web crawling technique is used to retrieve text data from WSJ.com and BBC.com.
2. Data Pre-processing: Then, Natural Language Processing is used to pre-process raw text data with less meaningless, noise words.
3. After that, name entity recognition is conducted on training data as one of the additional features to improve prediction accuracy.
4. Predictive Training: Last but not the least, LSTM (Long-short term memory) is trained and tested on post-processed data.

2. Motivation

Digital media is one of the best few places from which we can extract valuable public emotion, ideas and attitudes towards celebrities, business entities or social phenomena. As the functions of social media and news platform (such as Twitter, Facebook) develop, people can not only express their ideas on those platforms in short text message, but can also show their attitudes towards a certain phenomenon by simply choosing positive/negative or by adding emoticons in their text. Similarly, for news data, we can manually attach sentiment variable to each news articles to create our own data set.

For financial analysts, it will be of great help for their investment decision making if we can do textual analysis on those text on a specific company, sector or desired dimension of relevance. In order to accomplish this, we are applying Natural Language Processing (NLP), within the Python framework to derive meaningful quantitative values for modelling. Recent researches have confirmed a correlation between textual sentiment and near horizon volatility.

One caveat about the result is that correlation does not imply causation. The flexibility of the level of unique interaction will need to be determined as per the level of comfort from the analyst themselves.

3. Data Acquisition

All of the data from this are scraped from WSJ.com and BBC.com using python package selenium accompanied with driver 'Chromedriver' to interface with browser. Only news articles from July 2017 to August 2017 are considered in here. The packages used for building connection to web browser are partially shown below:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

And then use keywords to locate specify articles by resource, data period and class name.

```
browser.implicitly_wait(5)
clear = browser.find_element_by_class_name('search-clr').click()
browser.find_element_by_xpath('//*[@text()="SEARCH"]').click()

menuOptions = browser.find_element_by_class_name('datePeriod')
browser.find_element_by_name("sfrom").send_keys("2017/08/02")
browser.find_element_by_name("sto").send_keys("2017/08/02")

browser.find_element_by_link_text("WSJ Blogs").click()
browser.find_element_by_link_text("WSJ Videos").click()
```

```
browser.find_element_by_link_text("WSJ Site Search").click()
```

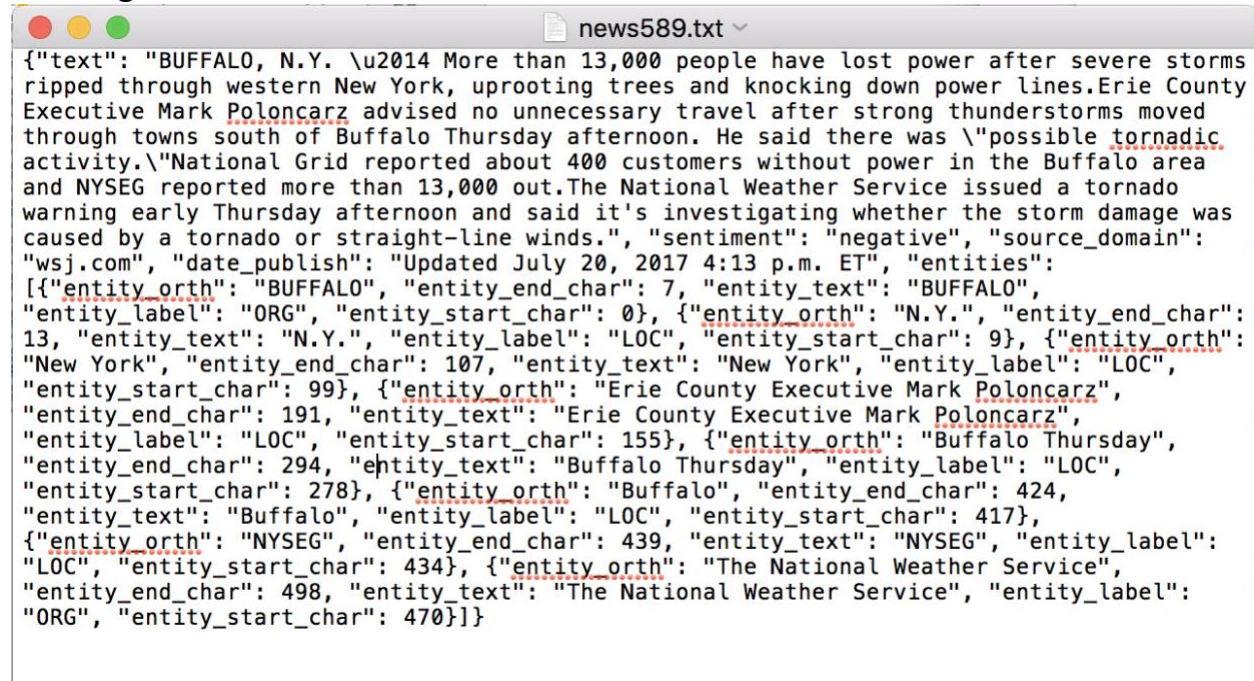
```
browser.execute_script("window.scrollTo(0, 0)")
```

```
searchArchive = browser.find_element_by_class_name('keywordSearchBar')
```

```
searchArchive.find_element_by_class_name("searchButton").click()
```

Several variables are retrieved for each news article: Text, Headline, No. id, Date Publish, Outlet(source). Those variables may or may not be included in the final model, but they are all critical characters to define a unique text.

The original data looks like this:



```
{
  "text": "BUFFALO, N.Y. \u2014 More than 13,000 people have lost power after severe storms ripped through western New York, uprooting trees and knocking down power lines.Erie County Executive Mark Poloncarz advised no unnecessary travel after strong thunderstorms moved through towns south of Buffalo Thursday afternoon. He said there was \"possible tornadic activity.\"National Grid reported about 400 customers without power in the Buffalo area and NYSEG reported more than 13,000 out.The National Weather Service issued a tornado warning early Thursday afternoon and said it's investigating whether the storm damage was caused by a tornado or straight-line winds.",
  "sentiment": "negative",
  "source_domain": "wsj.com",
  "date_publish": "Updated July 20, 2017 4:13 p.m. ET",
  "entities": [
    {
      "entity_orth": "BUFFALO",
      "entity_end_char": 7,
      "entity_text": "BUFFALO",
      "entity_label": "ORG",
      "entity_start_char": 0
    },
    {
      "entity_orth": "N.Y.",
      "entity_end_char": 13,
      "entity_text": "N.Y.",
      "entity_label": "LOC",
      "entity_start_char": 9
    },
    {
      "entity_orth": "New York",
      "entity_end_char": 107,
      "entity_text": "New York",
      "entity_label": "LOC",
      "entity_start_char": 99
    },
    {
      "entity_orth": "Erie County Executive Mark Poloncarz",
      "entity_end_char": 191,
      "entity_text": "Erie County Executive Mark Poloncarz",
      "entity_label": "LOC",
      "entity_start_char": 155
    },
    {
      "entity_orth": "Buffalo Thursday",
      "entity_end_char": 294,
      "entity_text": "Buffalo Thursday",
      "entity_label": "LOC",
      "entity_start_char": 278
    },
    {
      "entity_orth": "Buffalo",
      "entity_end_char": 424,
      "entity_text": "Buffalo",
      "entity_label": "LOC",
      "entity_start_char": 417
    },
    {
      "entity_orth": "NYSEG",
      "entity_end_char": 439,
      "entity_text": "NYSEG",
      "entity_label": "LOC",
      "entity_start_char": 434
    },
    {
      "entity_orth": "The National Weather Service",
      "entity_end_char": 498,
      "entity_text": "The National Weather Service",
      "entity_label": "ORG",
      "entity_start_char": 470
    }
  ]
}
```

4. Data Pre-Processing

4.1 Natural Language Processing

For every readable, grammatically errorless language, there will always exist special characters that convey little or no meaningful semantic idea, which do no help to

textual analysis and thus need to be removed. Here are all kinds of adjustment conducted on raw text and the corresponding Python functions:

- a. Contraction expansion
- b. Stop words
- c. Upper cases to lower cases
- d. Name entities recognition
- e. Special symbols and punctuations

Here are two examples of script functions to remove symbols, punctuations and stop words.

```
def nlp_remove_stop_words(self):
    test = bds_nlp_utils(self.input_text).nlp_contraction_expansion().split(" ")
    result = ' '.join([w for w in test if w.lower() not in spacy.en.STOP_WORDS])
    return result
```

```
def nlp_remove_symbols_punct(self,
                              filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                              lower=False, split=" "):
    string = bds_nlp_utils(self.input_text).nlp_contraction_expansion()
    text = string.translate(maketrans(filters, split * len(filters)))
    seq = text.split(split)
    symbols_punctuation = ' '.join([i for i in seq])
    return symbols_punctuation
```

4.2 Name Entity

Having reliable and valuable attributes is the key to successful model training and deep learning. Name entity should be one of the most meaningful features in this project because we are looking for features that represent business entities/individuals as well as reflect current popular financial spots. The people who is gaining social attention (For example, Trump and Hillary during the 2017 Presidential Election), the company which is developing new creation, or a city that

recently suffered from unpredictable disaster, should all be marked as special focuses. Therefore, name entity recognitions tags are manually added to each news articles through a training UI.

After manually adding name entity recognition, each observation now has more information that might imply its theme and category.

4.3 Special Focus on Sentiment Variable

Initially the sentiment variable of each article is tagged using Microsoft Azure Text Analytics API. The API accepted text contents and returned a sentiment score ranging from 0 to 1. Score near 0 is indicated to be negative and scores near 1 is indicated to be positive. However, in this project, since we were going to build a similar sentiment analytical tool, the input we need to train the model should have binary or ternary sentiment labels. Therefore, when we processed the sentiment score from the API, we took articles with sentiment scores smaller than 0.5 as negative and gave them a new sentiment value 1. For those with scores greater than 0.5 we gave them sentiment value 3. This is the original sentiment attribute of our data set.

Though the sentiment scores come from the analysis of Microsoft sentiment tool which is to some extent reliable, to elevate the reliability to a higher level, another sentiment variable is added manually by me after quickly reading each article. The value is either 'positive' or 'negative'. This is marked as 'sentiment' in our final data set and will be put into further analysis as target prediction label.

5. Data Exploration

5.1 Divide data set into Training/Testing data

We divided all data to be 80% training data and 20% testing data, each article are saved as a single .txt file and put in the folder called 'neg' or 'pos' according to their 'sentiment' label.

Here is the script used to divide data into training and testing set.

```
queryUI_select_Tagged= """
SELECT c.id,c.text, c.training_sentiment, c.sentiment, c.entities, c.last_updated, c.date_publish
FROM c
WHERE is_defined(c.entities) and is_defined(c.training_sentiment) and
is_defined(c.last_updated) and c.last_updated >= 1502719211
order by c.last_updated desc
"""
```

```
def Remove_duplicate(list, ignore_keys):
    seen = set()
    for row in list:
        index = frozenset((k,v) for k,v in row.items() if k not in ignore_keys)
        if index not in seen:
            yield row
            seen.add(index)
```

```
null_count=0
neg_list = list()
pos_list = list()
for item in rawnews_list:
    if item['training_sentiment']=='negative':
        print(item['text'])
        neg_list.append(item)
    elif item['training_sentiment']=='positive':
        print(item['text'])
        pos_list.append(item)
    else:
```



```
null_count+=1
```

```
pass
```

5.2 Simple glance

For most machine learning practices, one of the most crucial parts is feature hashing. Here string text is transformed to array of numbers to fit in to models. The feature hashing method used in this project is to get the ranking of all meaningful tokens for each article as final training data set. The feature-hashed result is a matrix of vectors with rankings of unigrams in each article. Length of feature vectors are limited to variable 'max_length' which is set to 400.

The result matrix looks like:

```
-----
Parsing texts...
train_docs: <class 'list'> 1304
[[597096   1340   5006 ...,    0    0    0]
 [716803   2688   7311 ...,    0    0    0]
 [  2268    205    709 ...,    0    0    0]
 ...,
 [107976    766   2555 ...,    0    0    0]
 [   827    375   1341 ...,    0    0    0]
 [  1560    160   1216 ...,    0    0    0]] <class 'numpy.ndarray'> 1304
-----
```

6. Predictive Model Training

6.1 LSTM Model building

As a special kind of recurrent neural network, Long-short Term Memory Model is capable of learning long term dependencies. There is a repeating module with four interactive layers. Structure:

Principle and algorithm Memory cell: A cell state to manage old and new information in the network, includes input gate, a neuron of self-recurrent connection, forget gate and an output gate.

The LSTM model consists of four layers: a dense layer, an embedding layer, a bidirectional layer and 64 hidden layers. Other essential variables are pre-set as following:

No. Hidden Layers	64	Text length	1000
No. Epoch	5	Entity length	30
Batch size	16	Drop-out ratio	0.5
Learning rate	0.001		

6.2 Model Training and Validation

Using Tensorflow as backend, LSTM model does the training on training data (Feature matrix with sentiment labels) and test the results with testing data. Each time it passes half of a batch of observation, it calculates the loss, accumulative accuracy. Once finish an epoch, it returns the overall loss and accuracy.

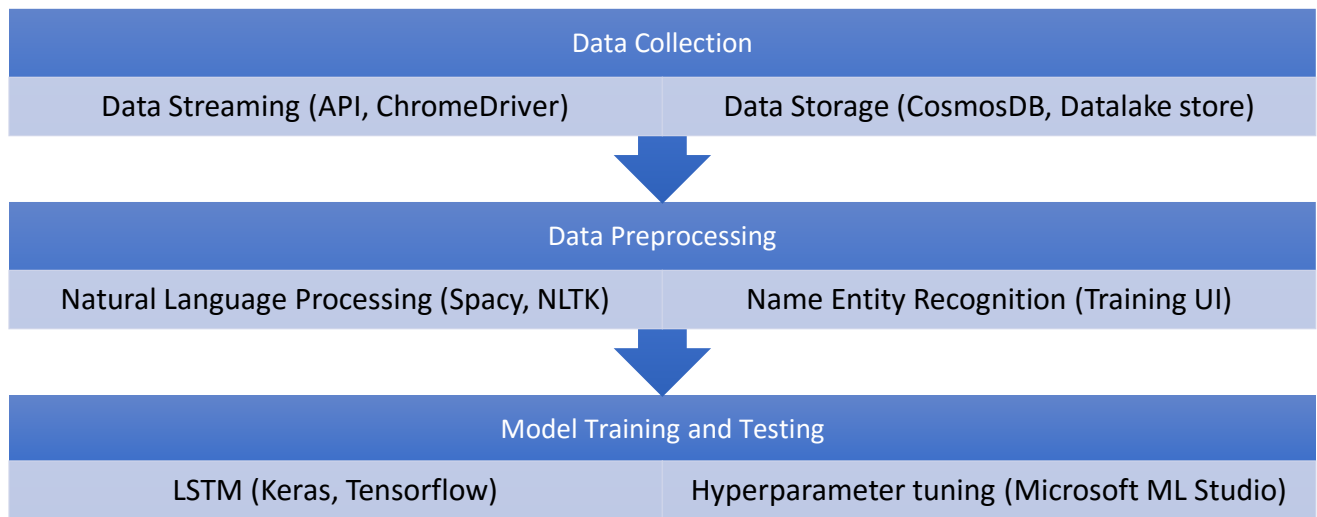
The result from LSTM with the above settings gives out an accuracy of 0.7270. Total training time is 2621s. This is a nice result for text analytics with such voluminous content, simply not only because text data is not designed to be as regular and meaningful as other experimental data like gene sequence or movie ratings, but also because it is the only information that can be made used for training.

```
Epoch 5/5
32/1304 [.....] - ETA: 200s - loss: 0.3058 - acc: 0.9062
48/1304 [>.....] - ETA: 94228s - loss: 0.3667 - acc: 0.8542
96/1304 [=>.....] - ETA: 45407s - loss: 0.3653 - acc: 0.8333
192/1304 [====>.....] - ETA: 21039s - loss: 0.4002 - acc: 0.8073
304/1304 [=====>.....] - ETA: 12007s - loss: 0.4180 - acc: 0.7961
320/1304 [=====>.....] - ETA: 22283s - loss: 0.4205 - acc: 0.7937
384/1304 [=====>.....] - ETA: 17385s - loss: 0.4375 - acc: 0.7839
464/1304 [=====>.....] - ETA: 13175s - loss: 0.4276 - acc: 0.7866
576/1304 [=====>.....] - ETA: 9220s - loss: 0.4396 - acc: 0.7830
592/1304 [=====>.....] - ETA: 8777s - loss: 0.4296 - acc: 0.7889
656/1304 [=====>.....] - ETA: 10769s - loss: 0.4160 - acc: 0.7973
768/1304 [=====>.....] - ETA: 7627s - loss: 0.4089 - acc: 0.7995
848/1304 [=====>.....] - ETA: 5884s - loss: 0.4131 - acc: 0.7925
864/1304 [=====>.....] - ETA: 7403s - loss: 0.4139 - acc: 0.7917
912/1304 [=====>.....] - ETA: 6252s - loss: 0.4198 - acc: 0.7917
928/1304 [=====>.....] - ETA: 5897s - loss: 0.4237 - acc: 0.7888
1008/1304 [=====>.....] - ETA: 4278s - loss: 0.4355 - acc: 0.7778
1104/1304 [=====>.....] - ETA: 2642s - loss: 0.4354 - acc: 0.7817
1120/1304 [=====>.....] - ETA: 2987s - loss: 0.4353 - acc: 0.7821
1152/1304 [=====>.....] - ETA: 2400s - loss: 0.4352 - acc: 0.7839
1184/1304 [=====>.....] - ETA: 1844s - loss: 0.4346 - acc: 0.7863
1200/1304 [=====>.....] - ETA: 1577s - loss: 0.4329 - acc: 0.7875
1296/1304 [=====>.....] - ETA: 112s - loss: 0.4316 - acc: 0.7886
1304/1304 [=====] - 18235s - loss: 0.4313 - acc: 0.7891 - val_loss: 0.5770 - val_acc: 0.7393
```

7. Conclusion

7.1 Project summary

The process of this project in this report can be described using the following graph.



To sum up, this is the data pulling and analytic part of a real-world industry project. Instead of taking clean data for one-time analysis, data is streamed and stored in cloud database before it is pulled again for further analysis. As for data preprocessing, standard NLP techniques is applied. The LSTM model gives out a prediction accuracy of over 70% which delivers a possible example for sentiment analysis in business and financial industry.

7.2 Future adjustment

- Consider neutral to be one of the sentiment value because there is no way preventing us from ignoring neutral attitudes which might be a sign of not popular or controversy on the target objects.
- Consider other deep learning models such as Controversial neural network(CNN), Gated Recurrent Neural Network(GRU) and combination of LSTM and RNN.
- Consider applying the project to cloud service for third party applications.

8. Reference

Packages and tools:

<https://pypi.python.org/pypi/selenium>

<https://sites.google.com/a/chromium.org/chromedriver/>

<https://github.com/explosion/spaCy>

<https://deeplearning4j.org/lstm.html>

<https://keras.io>

Concepts and discussions:

<https://github.com/andrewt3000/DL4NLP>

<http://runder.io/deep-learning-nlp-best-practices/>

<https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/>

<https://github.com/adeshpande3/LSTM-Sentiment-Analysis>

<https://towardsdatascience.com/sentiment-analysis-using-rnns-lstm-60871fa6aeba>

9. Appendix

For sample data set and Python scripts for model training please refer to the attached files.

Data set: folder 'news_data'

Model training: LSTM_training.py