

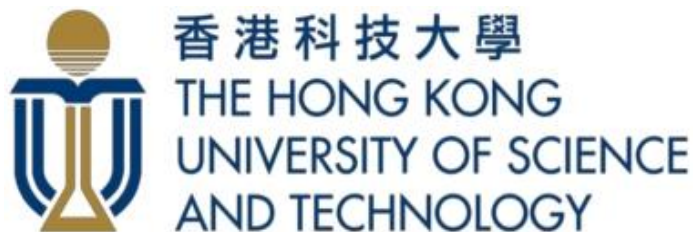
Stock Prediction Using Natural Language Processing

Master of Science in Big Data Technology

Submitted by

Haojun Chen
Jingyu Cui
Yuqing Huang
Zihao Peng

(Arranged in alphabetical order)



Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Spring Semester 2019

Contents

1	Introduction	2
1.1	System Architecture and Technology	2
2	Data Collection and Wrangling	3
3	Workflow	4
3.1	Kafka	4
3.2	NLP	6
3.2.1	CountVectorize	7
3.2.2	TF-IDF	8
3.2.3	Word2Vec	8
3.3	Prediction	9
3.3.1	<i>sklearn.naive_bayes.GaussianNB</i>	10
3.3.2	<i>pyspark.ml.classification.RandomForestClassifier</i>	10
3.3.3	<i>pyspark.mllib.classification.LogisticRegressionWithSGD</i>	11
3.4	Spark Streaming	11
3.5	Storage in MongoDB	12
4	Benefits of Using Big Data Technologies	14
4.1	RDD	14
4.2	DataFrame	14
4.3	MongoDB and Mongo-Spark-Connector	14
4.4	Kafka	15
4.5	Spark MLlib	15
4.6	Spark Streaming	15
5	Environment	15
6	Conclusion	15

Abstract

We used Machine learning techniques to evaluate past data pertaining to the stock market and world affairs of the corresponding time period, in order to make predictions in stock trends. We built a model that will be able to buy and sell stock based on profitable prediction, without any human interactions. The model uses Natural Language Processing (NLP) to make smart “decisions” based on current affairs, article, etc. With NLP and the basic rule of probability, we aim to achieve the goal of increasing the accuracy of the stock predictions. In order to fit the practical scenario, we use kafka and Spark Streaming, Based on these and real-time news information we can analyze and predict the stock movements on real-time. Considering the large amount of news data in real life, we implement a distributed database MongoDB

1 Introduction

Natural Language Processing is a technique used by a computer to understand and manipulate natural languages. With the help of natural languages, we manage all human derived languages. In short, natural language processing or NLP is used to analyze text and let machines derive meaning from the input. This human-computer interaction allows us to come up with many different applications to bring man and machine as one. For example, Google Translation and speech recognition are NLP techniques. In our project, we make use of some established NLP techniques together with Spark SQL to evaluate past data pertaining to the stock market and world affairs of the corresponding time period, in order to make predictions in stock trends.//

In order to proceed with this objective, we need to understand what Sentimental Analysis is. It is an analytical method that the computer uses to understand a natural language and deduce if the message is positive, neutral or negative. In our project, Sentimental Analysis refers to the deduction of the news headlines on whether they regard the stock price will increase or decrease. By doing so, we end up with the ‘emotional’ status of the data, which is what sentimental analysis gives to its user.

1.1 System Architecture and Technology

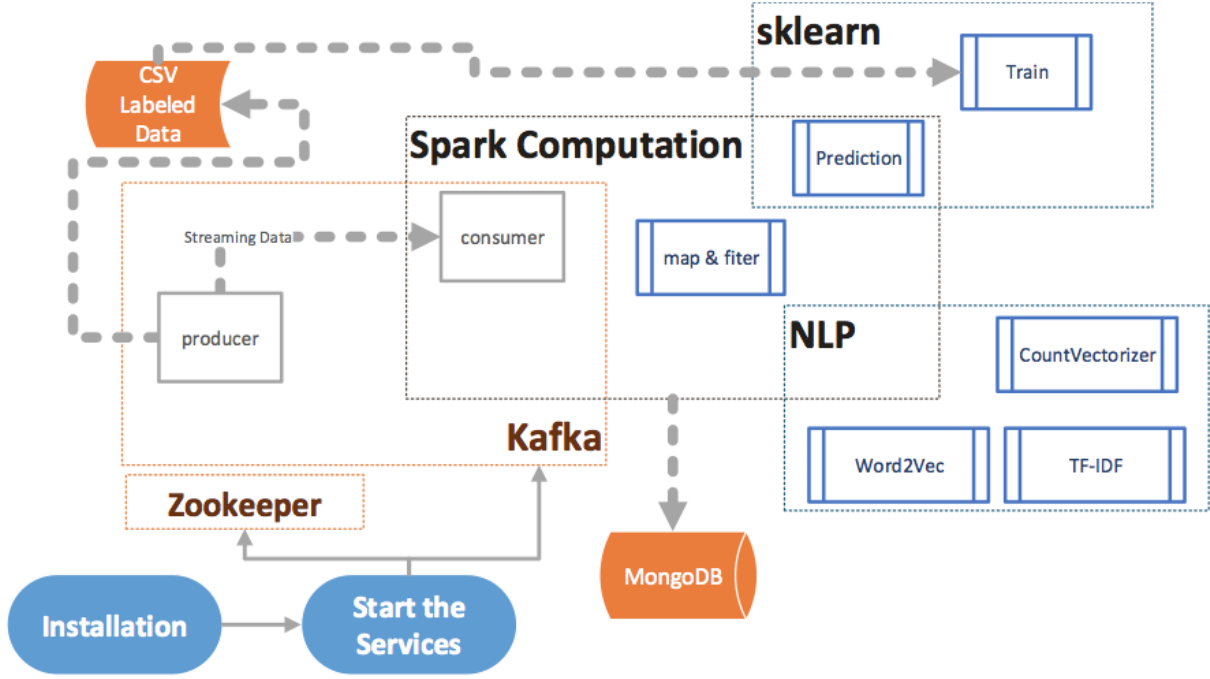


Figure 1: System Architecture

2 Data Collection and Wrangling

We have used the *CombinedNews_{DJIA}.csv* dataset (courtesy to author Aaron7sun on Kaggle.com). The *CombinedNews_{DJIA}.csv* dataset spans from 2008 to 2016. We extend the dataset to include additional data. This additional data is collected from the Guardian’s Restful News API for the 2000 - 2008 period. We take the 25 most popular headlines for each given day in this period. In addition, we pull the Dow Jones Index (DJI) of Yahoo Finance’s website for the 2000 - 2008 period to compare the influences of data. There are two channels of data provided in this dataset:

News data that has historical news headlines from Reddit World News Channel. Only the top 25 headlines are considered for a single date.

Stock data for Dow Jones Industrial Average (DJIA) over the corresponding time range is used to label the data. The stock data is compiled from Yahoo Finance.

Note: The headlines for each dataset acts as explanatory data that causes the stock price to either rise (labeled 1) or to fall (labeled 0). We have the top 25 headlines for one single date arranged as one row of the extracted data set.

Since our goal is to predict the tendency of the stock of a specific company, the data that lead the stock’s price of the next day to decline or stay the same are labelled “0”, while the data that lead the price of the next day to rise are labelled “1”. We compare the data between the two pulled datasets and then merge them into one to get more accurate prediction.

With the raw data, we cannot proceed much further until we manipulate the data to suit our analysis and convert the data into vectors that are much easier to work on. For this, we use Word2Vec. This is a group of related models used to create word embeddings. Namely, word embeddings are sets of language modeling and feature learning

techniques in NLP where words or phrases from the vocabulary are mapped to vectors of real numbers. These vectors make up the training and test sets. English is really easy, all those spaces make it really easy to be tokenized – in other words, to determine what’s a word. So, we just use a simple set of rules for English tokenization.

Such raw data is manipulated using python. We first split the data into lists of words, but these lists are flooded with HTML tags and punctuations so we cleaned up the data and removed all HTML tags and punctuations. Then we moved forward with removing stop words. Stop words are words that do not contribute to the meaning or sentiment of the data such as ‘the’, ‘is’, ‘and’, etc. We have also converted all the letters to lowercase to create a more even data set to play with (Figure2).

Date	label	words
2008-08-08	0	[Georgia, downs, ...]
2008-08-11	1	[Why, wont, Ameri...]
2008-08-12	0	[Remember, that, ...]
2008-08-13	0	[refuses, Israel,...]
2008-08-14	1	[All, the, expert...]
2008-08-15	1	[Mom, of, missing...]
2008-08-18	0	[In, an, Afghan, ...]
2008-08-19	0	[Man, arrested, a...]
2008-08-20	1	[Two, elderly, Ch...]
2008-08-21	1	[British, residen...]
2008-08-22	1	[Syria, says, its...]
2008-08-25	0	[Korea, Kim, died...]
2008-08-26	1	[North, Korea, ha...]
2008-08-27	1	[Photos, of, 15, ...]
2008-08-28	1	[Military, help, ...]
2008-08-29	0	[Russian, Prime, ...]
2008-09-02	0	[girl, filmed, Is...]
2008-09-03	1	[Poland, Legaliza...]
2008-09-04	0	[Security, guards...]
2008-09-05	1	[In, Jordan, the,...]

Figure 2: Data Manipulation

3 Workflow

3.1 Kafka

We use Kafka to build a real-time streaming application that transforms or reacts to the streams of data. The following flow picture is a Kafka message transfer process.

We use Kafka to create a producer issue topic “test” so we can get a Real-time information flow.

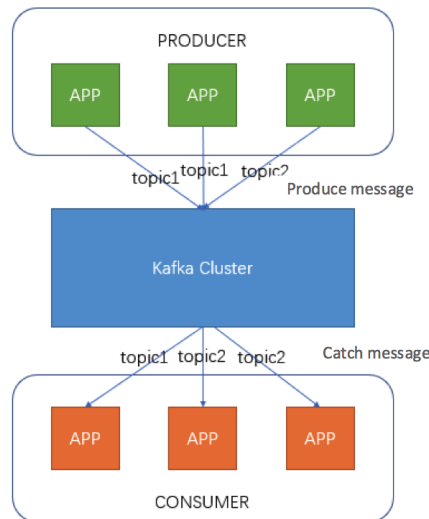


Figure 3: Workflow

```
def main():
    #create module
    producer = KafkaProducer(bootstrap_servers=['127.0.0.1:9092'])
    with open('Full_Data.csv','r',encoding = "UTF-8") as f:
        for line in f.readlines():
            time.sleep(1)
            print(type(line))
            producer.send("txt",line.encode('utf-8'))
            print (line)
    #producer.flush()
```

Figure 4: Kafka Client

```
<class 'str'>
2000/1/20,0,Homes alone won't make us world beaters,Wembley crackdown on pirat
e hospitality,Keegan calls for pay curb,Backing for Saint Jones,Leboeuf rises
to Bates,South Africa Test averages,Wasim loses way after milestone,Swann aske
d to hang around,Rousing finale to an old refrain,Lords back trial by jury,Hol
lywood star Hedy Lamarr dies,Lords reform,Britain's ethical foreign policy: ke
eping the Hawk jets in action,The ?bn conflict of interest,Should you be inves
ting in your future?,Is postgraduate study for you?,Turn these clients away,Th
e generals profit as the people pay with their lives,The Wakeham commission re
commendations,Vanessa Feltz returns to daytime television,Tesco leads the worl
d online growth and profit,Cybersurfing: window shopping on the web,Bargain hu
nting: how to compare prices,Cybershopping's 50 favourite sites,Falling for V
ertigo by WG Sebald
```

Figure 5: Kafka Server

Then, we create a consumer in the spark application. It captures the information and transforms as spark streaming. For spark streaming we do same operation for each RDD.

```
sconf=SparkConf().setMaster("local[2]")
sconf.set('spark.cores.max',3)
sc=SparkContext(appName='txt',conf=sconf)
sameModel = SVMModel.load(sc, "SVMmodel")
kfk_brokers = {"bootstrap_servers": "127.0.0.1:9092",
               "kafka.bootstrap.servers": "127.0.0.1:9092",
               "brokers": "127.0.0.1:9092",
               "host": "127.0.0.1:9092"}

brokers = "127.0.0.1:9092,127.0.0.1:9092"
ssc=StreamingContext(sc,2)
# Set the number of batches to partition, which must be set.
#The execution frequency of batch processing depends on the demand
#and availability of cluster/server resources
user_data = KafkaUtils.createDirectStream(ssc,['txt'],kafkaParams={"metadata.broker.list":brokers})

user_fields = user_data.map(lambda line: line[1].split(','))


user_fields.pprint()

RandomForestmodel = joblib.load('RandomForest2.model')

user_fields.foreachRDD(lambda rdd: rdd.foreach(pre))

ssc.start()
ssc.awaitTermination()
```

Figure 6: Kafka Server



```
Time: 2019-05-25 15:43:36
-----
[ '2004/6/28', '0', 'Gerrard pledges future to Liverpool', 'The way you see it', 'Denmark and Tomasson depart too quietly', 'Euro 2004 in brief', 'Web heroine turns out to be hoax', 'Henman beats OPhilippoussis 6-2', '7-5', '6-7', '7-6', 'Henman v Philippoussis - live!', 'Ex-wife to testify in Billie-Jo appeal', 'Blair and Brown bring adviser to book', 'In a different league', 'Hugh Masekela', 'Barbican', 'London', 'Review of food available at Glastonbury', 'O'Reilly steps down as INM chair', 'Hewitt to face Federer', 'Shareholders vent anger over Shell scandal', 'Glastonbury fashion reviewed', 'Q&A: Marks & Spencer', 'M&C Saatchi founders set for 76m windfall', 'Jowell could halve BBC charter', 'Bush and Blair welcome sovereign Iraq', 'Guantanamo detainees granted access to US courts', 'Beware of bugs like this', 'Make crime our priority', 'urges IDS', 'Gerrard to stay at Anfield', 'Health choice plan condemned by doctors\n']
```

Figure 7: Kafka Server

3.2 NLP

Most of our data comes from the news. Natural language plays an important part, so we use different Natural Language Processing (NLP) methods to interpret and construct the data set. The data are composed of a row of sentences. In order to reduce the complexity, the stop words such as “a”, “and”, “the” have been cleaned.

Most of our data comes from the news. Natural language plays an important part, so we use different Natural Language Processing (NLP) methods to interpret and construct

the data set. The data are composed of a row of sentences. In order to reduce the complexity, the stop words such as “a”, “and”, “the” have been cleaned.

The thing is that we cannot stop at just using the Bag of Words model as this generates feature vectors that only give importance to the number of occurrences of words rather than where they occur and with what words they accompany. To resolve this, we use the n-gram model and the skip gram model. With either models, we can store the order of words in the way they occur in the data. The number of words stored in a single order depends on the value on n. Say n=2, calls for a bigram model, which stores sets of 2 words in order.

In addition, we find that the N-gram model helps predict the next set of words in an n- worded text or speech. Because of this, we consider discrete-time Markov chain and Word2Vec model. Considering the model efficiency (because we want to use this in the Real-time analysis), we choose word2Vec model.

3.2.1 CountVectorize

CountVectorize is a common feature of the numerical calculation class - a text feature extraction method. For each training text, it only considers the frequency of each word in the training text.

The words in the text will be converted into word frequency matrix, which calculates the occurrence times of each word through the fit transform function.

'a	hindrance	to	operations	extracts	from	the	leaked	report	(0, 41378)	1
s	scorecard	hughes	instant	hit	buoys	blues	jack	gets	(0, 33409)	1
tes	on	at	ice	cold	alex	chaos	as	maracana	(0, 29186)	1
builds	up	for	unite	d	depleted	leicester	prevail	as	(0, 38097)	1
elliott	spoils	everton	s	part	y	hungry	spurs	sense	(0, 35042)	1
rich	pickings	gunners	so	wide	of	an	easy	target	(0, 35266)	1
derby	raise	a	glass	to	strupar	s	debut	double	(0, 15203)	1
southgat	e	strikes	leeds	pay	the	penalty	hammers	hand	(0, 14929)	1
robson	a	youthf	ul	lesson	saints	party	like	it	(0, 28934)	1
s	wear	wolves	have	turned	into	lambs	stump	mike	(0, 6516)	1
catches	testy	gough	s	taunt	langer	esca	pes	to	(0, 34178)	1
hit	flintoff	injury	piles	on	woe	for	england	hunte	(0, 45623)	1
rs	threaten	jospin	with	new	battle	of	the	somme	:	:
succes	sor	drawn	into	scandal	the	difference	between	men	(0, 2191)	2
and	women	s	ara	denver	nurse	turned	solicitor	diana	(0, 6844)	1
s	landmine	crusade	p	ut	tories	in	a	panic	(0, 1024)	1
yeltsin	s	resignation	caught	opposition	flat	footed	russian	roulette	(0, 7817)	1
sold	out	recovering	a	title'					(0, 19404)	1
									(0, 2414)	1
									(0, 28826)	2
									(0, 37516)	1
									(0, 18626)	1
									(0, 16333)	1
									(0, 20936)	1
									(0, 4498)	1
									(0, 5650)	1
									(0, 18643)	2
									(0, 20308)	1
									(0, 19187)	1

Figure 8:

In our research and study, the length range of phrase segmentation divides the adjacent words used to calculate word frequency are also influenced with the model precision rate. For example, we can observe this in the case of RandomForest Classifier algorithm.

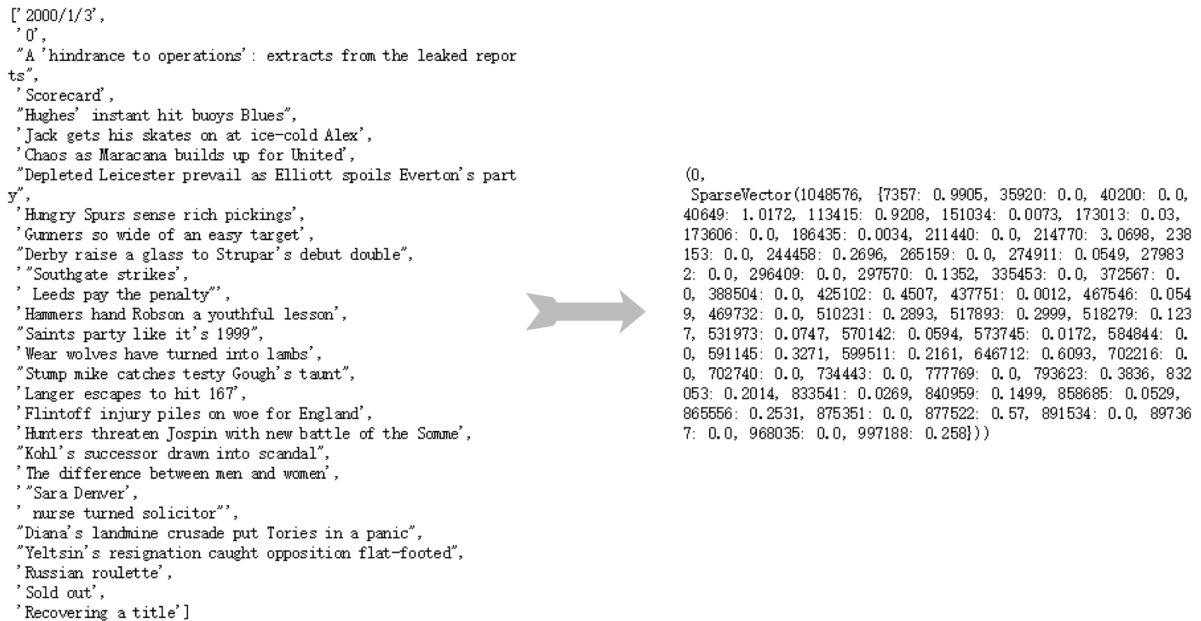
Table 1: Random Forest

ngram range	category	precision	recall	f1 score
(1, 1)	0	0.89	0.75	0.81
(1, 1)	1	0.79	0.91	0.85
(1, 2)	0	0.94	0.72	0.81
(1, 2)	1	0.78	0.95	0.86
(2, 3)	0	0.97	0.72	0.82
(2, 3)	1	0.78	0.98	0.87

3.2.2 TF-IDF

TF-IDF (Term frequency-inverse document frequency), a common weighted technique for information retrieval and information exploration. TF-IDF is a statistical method used to evaluate the importance of a word to a set of documents or to one of the documents in a corpus. The importance of a word increases directly with the number of times it appears in a document, but at the same time it decreases inversely with the frequency of its appearance in the corpus.

We use this function with `mllib`. This function is much more simple and quick.



```

[' 2000/1/3',
 '0',
 'A 'hindrance to operations': extracts from the leaked repor
ts',
 'Scorecard',
 'Hughes' instant hit buoys Blues',
 'Jack gets his skates on at ice-cold Alex',
 'Chaos as Maracana builds up for United',
 'Depleted Leicester prevail as Elliott spoils Everton's part
y',
 'Hungry Spurs sense rich pickings',
 'Gunners so wide of an easy target',
 'Derby raise a glass to Strupar's debut double',
 'Southgate strikes',
 'Leeds pay the penalty',
 'Hammers hand Robson a youthful lesson',
 'Saints party like it's 1999',
 'Wear wolves have turned into lambs',
 'Stump mike catches testy Gough's taunt',
 'Langer escapes to hit 167',
 'Flintoff injury piles on woe for England',
 'Hunters threaten Jospin with new battle of the Somme',
 'Kohl's successor drawn into scandal',
 'The difference between men and women',
 'Sara Denver',
 'nurse turned solicitor',
 'Diana's landmine crusade put Tories in a panic',
 'Yeltsin's resignation caught opposition flat-footed',
 'Russian roulette',
 'Sold out',
 'Recovering a title']

```

→

```

(0,
 SparseVector(1048576, {7357: 0.9905, 35920: 0.0, 40200: 0.0,
40649: 1.0172, 113415: 0.9208, 151034: 0.0073, 173013: 0.03,
173606: 0.0, 186435: 0.0034, 211440: 0.0, 214770: 3.0698, 238
153: 0.0, 244458: 0.2696, 265159: 0.0, 274911: 0.0549, 27983
2: 0.0, 296409: 0.0, 297570: 0.1352, 335453: 0.0, 372567: 0.
0, 388504: 0.0, 425102: 0.4507, 437751: 0.0012, 467546: 0.054
9, 469732: 0.0, 510231: 0.2893, 517893: 0.2999, 518279: 0.123
7, 531973: 0.0747, 570142: 0.0594, 573745: 0.0172, 584844: 0.
0, 591145: 0.3271, 599511: 0.2161, 646712: 0.6093, 702216: 0.
0, 702740: 0.0, 734443: 0.0, 777769: 0.0, 793623: 0.3836, 832
053: 0.2014, 833541: 0.0269, 840959: 0.1499, 858685: 0.0529,
865556: 0.2531, 875351: 0.0, 877522: 0.57, 891534: 0.0, 89736
7: 0.0, 968035: 0.0, 997188: 0.258}))

```

Figure 9: TF-IDF

3.2.3 Word2Vec

The Google's Word2Vec deep learning method is also provided to focus on the sentiment of the words by means of the bag of words concept. This method is suitable for us because it doesn't require labels in order to create meaningful representations. If there are enough training data, it would produce word vector with intriguing characteristics

so that we could analyze the relationship between the words that have similar meanings. The word2vec model is a simplified neural network.

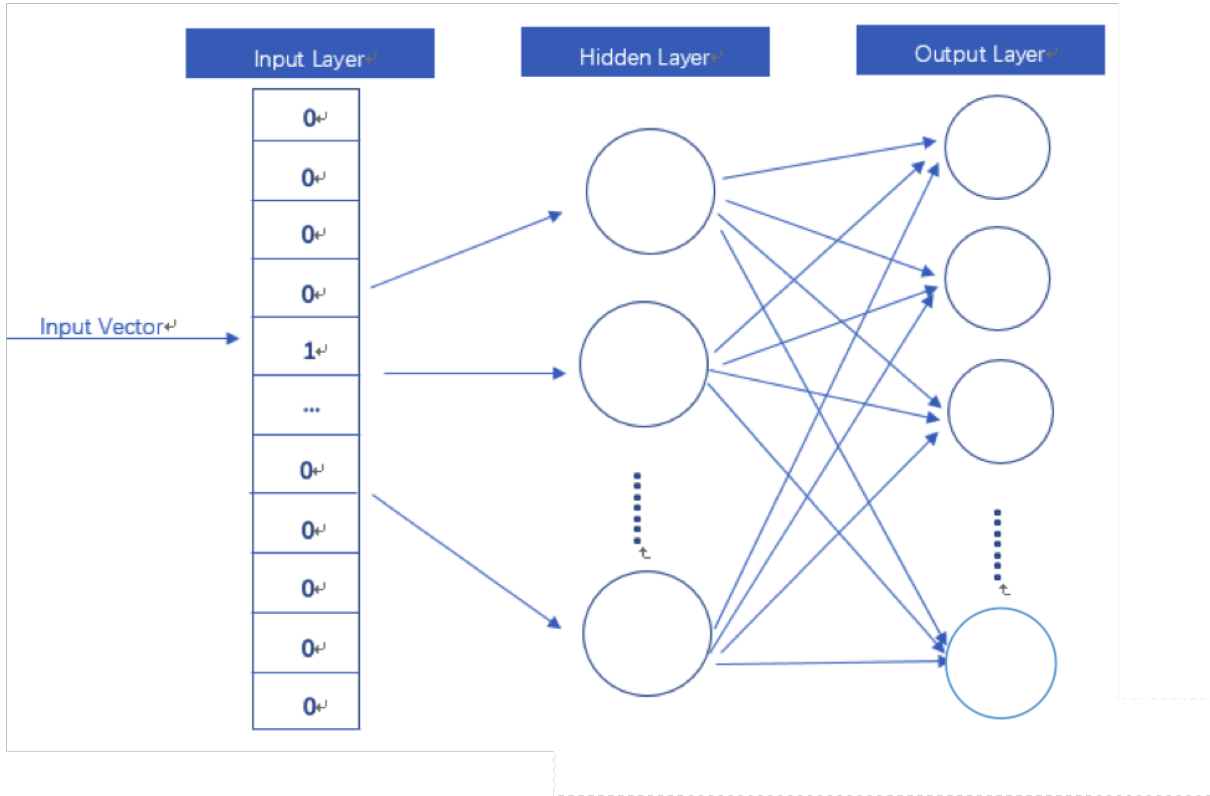


Figure 10: Structure of Word2Vec Model

The input is a one-hot vector. Hidden Layer does not activate the function, which is the linear unit. The Output Layer dimension is the same as the Input Layer dimension, and Softmax regression is used too. The dense vector we want to get is actually the output unit of the Hidden Layer.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix} \quad (1)$$

Here, we care about the weight of the parameters in the first hidden layer of the model, and the parameter matrix is the word vector we need. Each of its rows is the word vector for the corresponding word in the dictionary, and the number of rows is the size of the dictionary.

3.3 Prediction

With all this done, we have our manipulated data vectors ready to be trained and tested. We have split the extracted dataset in the ratio of 4:1. 80% of the extracted data will be the training data and 20% of the extracted data will be the test data.

Table 2: Random Forest				
	precision	recall	f1 score	support
0	0.89	0.76	0.82	182
1	0.79	0.91	0.85	192
<i>micro avg</i>	0.83	0.83	0.83	378
<i>macro avg</i>	0.84	0.83	0.83	378
<i>weighted avg</i>	0.84	0.83	0.83	378

In this process, we use three different Machine learning algorithm libraries including sklearn, pyspark.mllib and pyspark.ml. These three libraries are also corresponding to three different data structures. These algorithms have the same algorithm in the database but their efficiency and usage vary based on different data structures and application scenarios.

Sklearn is an open source library for machine learning in python, through which machine learning algorithms can be easily invoked to complete practical tasks. However, it can only run on a single machine. This is very inefficient when faced with a large scale of data and computing clusters.

Pyspark.mllib is based on RDD (Resilient Distributed Dataset, is the most basic data abstraction in Spark), a distributed implementation of some machine learning algorithms is provided, which can process data in parallel and has good efficiency for computing clusters and large-scale data sets.

For pyspark.ml, Spark officially recommends using ml, because ml is more comprehensive and flexible, and will mainly support ml in the future. Mllib is likely to be discarded. ML mainly operates on DataFrame, while mllib operates on RDD, which means that the data sets targeted by the two are different. ML has a higher level of abstraction over DataFrame and a lower degree of data and operation coupling than the basic operations mllib provides on RDD. Operations in ml can be pipelined. Similar to sklearn, many operations (algorithm/feature extraction/feature conversion) can be strung together as pipelines and data can flow through the pipeline.

3.3.1 *sklearn.naive_bayes.GaussianNB*

In this library we select GaussianNB. GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}} \quad (2)$$

And the result is:

3.3.2 *pyspark.ml.classification.RandomForestClassifier*

Random forest is the integrated algorithm of decision tree. The random forest contains multiple decision trees to reduce the risk of overfitting. Random forest is also easy to interpret and is capable of handling category features, and is easy to expand to multi-classification problems, without the requirement of feature scaling.

Table 3: Random Forest				
	precision	recall	f1 score	support
0	0.94	0.72	0.81	186
1	0.78	0.95	0.86	192
<i>avg/total</i>	0.85	0.84	0.83	378

Random forest trains a series of decision trees respectively, so the training process is parallel. Each decision tree is slightly different because a random process is added to the algorithm. By Merging the predicted results of each tree, the variance of the prediction is reduced and the performance on the test set is then improved.

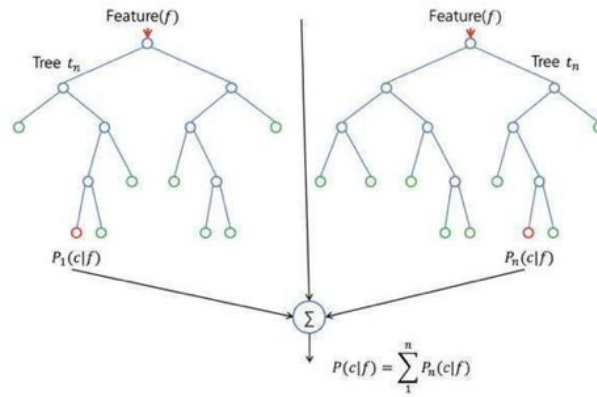


Figure 11: Random Forest

And the result is:

3.3.3 `pyspark.mllib.classification.LogisticRegressionWithSGD`

Logistic regression is essentially a linear regression, except that a layer of function mapping is added in the mapping from feature to result, that is, the linear sum of feature is taken first, and then the function $g(z)$ is used to make the prediction as the hypothesized function. $g(z)$ maps continuous values to 0 and 1.

It differs from linear regression in that in order to compress a large range of the output of linear regression, for example, from $-\infty$ to ∞ , to between 0 and 1, such output values expressed as "possibility" can convince the general public. Of course, the nice thing about compressing large values into this range is that it eliminates the effects of particularly sharp variables.

3.4 Spark Streaming

This is one of several libraries that the Spark platform provides (others include Spark SQL, Spark MLlib, and Spark GraphX). Spark Streaming provides a way of processing

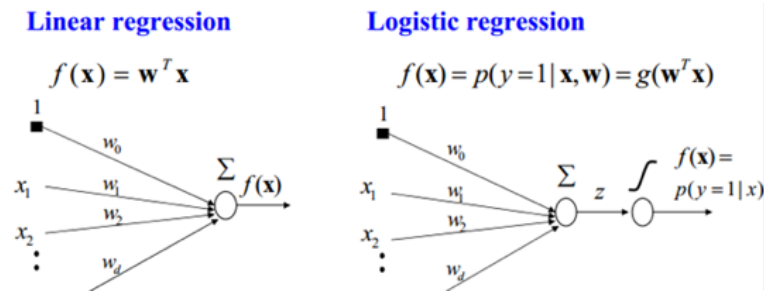


Figure 12: Logistic Regression

"unbounded" data - commonly referred to as "streaming" data. It does this by breaking it up into micro batches, and supporting windowing capabilities of processing across multiple batches.



Figure 13: Workflow of Spark Streaming

The use case I'm going to put together is - almost inevitably for a generic unbounded data example - using Twitter, read from a Kafka topic. We simply start counting the number of tweets per user within each batch and doing some very simple string manipulations.

3.5 Storage in MongoDB

MongoDB is a database management system designed for web applications and Internet infrastructure. We used MongoDB to store the stream data and prediction results we received from Kafka.

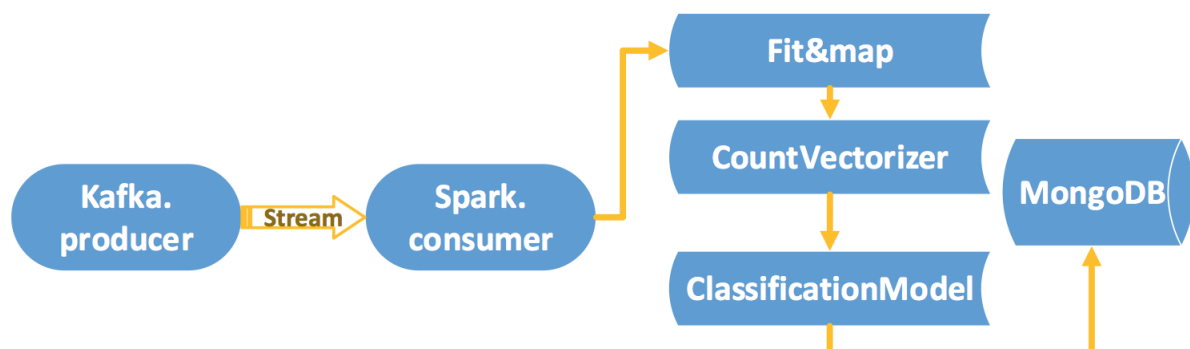
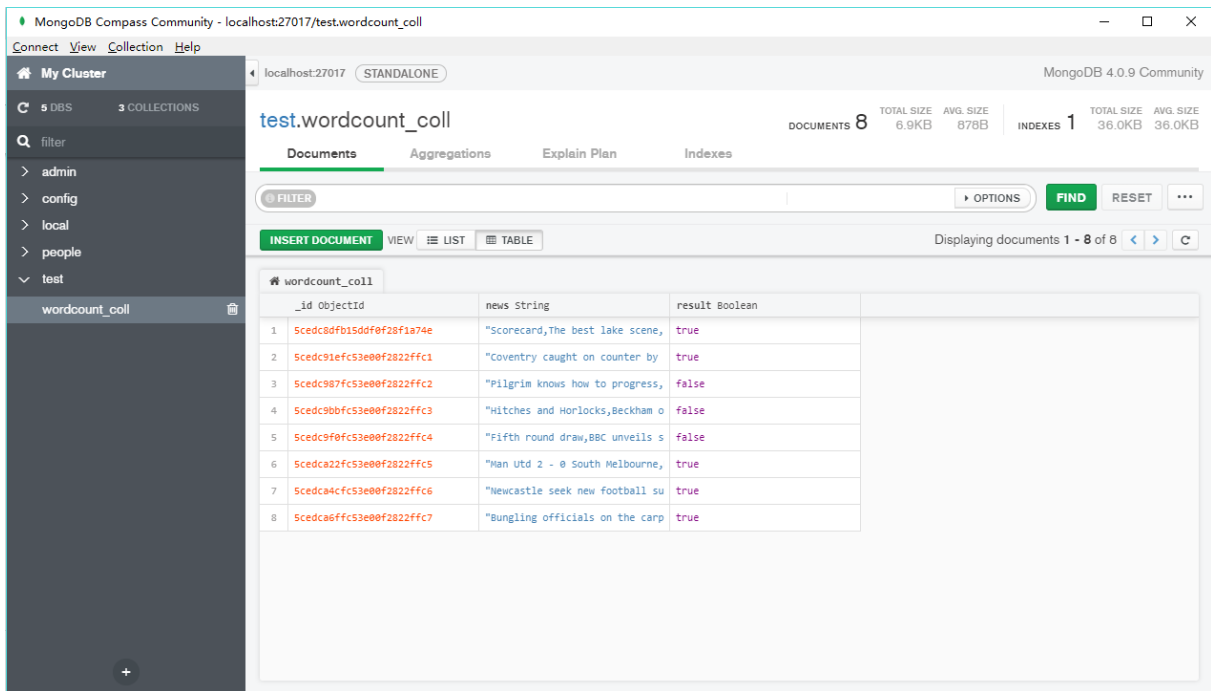


Figure 14: Workflow of MongoDB

In this section, we implemented MongoDB. The workflow of MongoDB is shown above. First, we connected MongoDB by specifying the test database, obtained the wordcount-coll set, and then we inserted the data in it by including their attributes such as results and news. We then terminated the connection. Next, we did the implementation of RDD, NLP and Kafka based on the MongoDB workflow. For example, we inserted the topic (txt) that the Kafka producer publishes - the topic that the Kafka consumer receives. The spark program first trains an NLP word vector model, and then performs a foreachrdd operation on the received dstream. In the rdd operation, we put each line of data into the word vector model to obtain the word vector and then put the word vector into trained random forest model for prediction, and finally put all the predictive values into MongoDB along with raw data. For more details, kindly refer to the `kafkaserver.py`.



MongoDB Compass Community - localhost:27017/test.wordcount_coll

Connect View Collection Help

My Cluster

localhost:27017 (STANDALONE)

MongoDB 4.0.9 Community

test.wordcount_coll

DOCUMENTS 8 TOTAL SIZE 6.9KB AVG. SIZE 878B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Explain Plan Indexes

FILTER

OPTIONS FIND RESET

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 8 of 8

	_id ObjectId	news String	result Boolean
1	Scedc8dfb15ddf0f28f1a74e	"Scorecard, The best lake scene,	true
2	Scedc91efc53e00f2822ffc1	"Coventry caught on counter by	true
3	Scedc987fc53e00f2822ffc2	"Pilgrim knows how to progress,	false
4	Scedc9bbfc53e00f2822ffc3	"Hitches and Horlocks, Beckham o	false
5	Scedc9f0fc53e00f2822ffc4	"Fifth round draw, BBC unveils s	false
6	Scedca22fc53e00f2822ffc5	"Man Utd 2 - 0 South Melbourne,	true
7	Scedca4cfc53e00f2822ffc6	"Newcastle seek new football su	true
8	Scedca6ffc53e00f2822ffc7	"Bungling officials on the carp	true

Figure 15: Main page of MongoDB

```

1 import sys
2 from pyspark import SparkConf
3 from pyspark import SparkContext
4 from pyspark.streaming import StreamingContext
5 from pyspark.streaming.kafka import KafkaUtils
6 import os
7 from pyspark.mllib.classification import SVMWithSGD, SVMModel
8 from pyspark.mllib.feature import HashingTF, IDF
9 from pyspark.mllib.regression import LabeledPoint
10 from sklearn.externals import joblib
11 from sklearn.feature_extraction.text import CountVectorizer
12 import pandas as pd
13 from pymongo import MongoClient
14
15 import os
16
17
18 def pre(rdd):
19
20     headlines=[]
21     headlines.append(' '.join(str(i) for i in rdd[2:-1]))
22     basictrain3 = basicvectorizer3.transform(headlines)
23
24     res=RandomForestmodel.predict(basictrain3)
25     connection = MongoClient()
26     test_db = connection.get_database('test')
27     wordcount_coll = test_db.get_collection('wordcount_coll')
28     wordcount_coll.insert({"result":res[0] ,"news":' '.join(str(i) for i in rdd[2:-1])})
29     connection.close()

```

4 Benefits of Using Big Data Technologies

4.1 RDD

For RDD (Resilient Distributed Datasets), the benefits of it are mainly from its features, including fault tolerance mechanism, distributed and in-memory storage. These features lead to a more robust and faster execution than MapReduce of Hadoop. Because of these features, RDD provides faster execution in iterative processing machine learning algorithms.

4.2 DataFrame

In Spark, DataFrame is a distributed data set organized by named columns, which are equivalent to tables in a relational database and data frames in R/Python (but with more optimization). Spark greatly optimizes more on the execution time and memory usage of DataFrame compared to RDD.

4.3 MongoDB and Mongo-Spark-Connector

MongoDB is a type of NoSQL database using JSON-like documents with schemas. It has user-friendly interface with sample statistical summary for each collection, MongoDB's data schema can be flexibly updated as the application evolves. At the same time, it provides developers with the functionality of a traditional database: secondary indexes, complete query systems, strict consistency, and so on. MongoDB can make enterprises more agile and scalable.

4.4 Kafka

Kafka is a high throughput for both publication and subscription. It is so powerful in that it is able to produce about 250,000 messages per second (50 MB) and process 550,000 messages per second (110 MB).

Persistent operations are possible. Messages are persisted to disk and are therefore available for bulk consumption, such as ETL, as well as real-time applications. Data loss is prevented by persisting the data to the hard disk and by replication. Distributed system, easy to extend to the outside have several producers, brokers and consumers all distributed. The machine can be extended without downtime.

4.5 Spark MLlib

Spark Machine Learning Library collects common feature selection, extraction method and general learning algorithm, which makes practical machine learning scalable and easy. Besides, the DataFrame based on pipeline structure supported by the library.

Spark officially recommends using ml, because ml is more comprehensive and flexible, and will mainly support ml in the future. Mllib is likely to be discarded. ML mainly operates on DataFrame, while mllib operates on RDD. This means that the data sets targeted by the two are different. ML has a higher level of abstraction over DataFrame while a lower degree of data and operation coupling than the basic operations that mllib provides on RDD as mentioned above.

MLlib provide following tools:

ML algorithm: general learning algorithms, such as classification, regression, clustering and collaborative filtering

Feature extraction, feature extraction, transformation, reduction and selection

Pipeline: a tool for building, evaluating, and tuning ML pipelines

Persistence: save and load algorithms, models, and pipes

4.6 Spark Streaming

Spark Streaming is a real-time computing framework based on Spark. Through its rich API and memory-based high-speed execution engine, users can combine Streaming, batch processing and interactive test query applications.

5 Environment

We used Jupyter Notebook, which is an open-source web application that allows us to create and share documents that contain live code, equations, visualizations and explanatory text. We perform data cleaning and transformation, statistical modeling and machine learning in this environment.

6 Conclusion

Social Media can sometimes be deceiving when delivering the right frame of speech. Here we have used twitter feed and news articles around the web that has influenced the

stock market of a company. Through this project, we have learnt the basics of Natural Language Processing. Even though you can't bet your money on the stock from this project, this work can be treated as solid understanding and application of the basics of Natural Language Processing. It is also feasible for us to use the same model for different text data. It was interesting to know about how to go from text data to vectors of numbers and how to apply Machine learning techniques that can help to influence the stock market of a company. It also helped us gain a wider sense of the power of NLP in various applications. From reading about machine learning models in class to implement them with real data and observe the performance of a model, tuning the parameters, performing exploratory data analysis set a great learning curve for future projects.

To get more insights on which model to use and how to construct them, we learnt by reading through research papers and usage of scikit learn to build our models. As any project that does not take a straight path towards completion, we hit certain roadblocks while implementing this project.

Road block 1:

As any machine learning task is concerned the data in consideration was restrictive and had few data cleaning to be done. Firstly, the data consisted of a character "b" appended to text in multiple ways and was a little challenging to remove it across the entire dataset.

Road block 2:

It was also challenging to find more data. The dataset available to use was from 2008 to 2016. We had to scrape it from another source (Yahoo News) completely and put in the format that we wanted to work for our Machine learning model. It was a challenging task to scrape and wrangle it to the data set that we wanted to (25 top headlines and labels associated with them)

Road block 3:

While we put the dataset for our SVM model, there were quite a few errors it kept throwing and one of them being NaN values and the number of cores we used to train our model. We used all the 4 cores to run our algorithm in parallel for faster execution.

Reference

- [1] L. Bing, K. C. C. Chan, and C. Ou. Public sentiment analysis in twitter data for prediction of a company's stock price movements. In *2014 IEEE 11th International Conference on e-Business Engineering*, pages 232–239, Nov 2014.
- [2] Z. Jiang, P. Chen, and X. Pan. Announcement based stock prediction. In *2016 International Symposium on Computer, Consumer and Control (IS3C)*, pages 428–431, July 2016.
- [3] M. Kaya, G. Fidan, and I. H. Toroslu. Sentiment analysis of turkish political news. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 174–180, Dec 2012.
- [4] D. Rao, F. Deng, Z. Jiang, and G. Zhao. Qualitative stock market predicting with common knowledge based nature language processing: A unified view and procedure. In *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pages 381–384, Aug 2015.
- [5] D. Sehgal and A. K. Agarwal. Sentiment analysis of big data applications using twitter data with the help of hadoop framework. In *2016 International Conference System Modeling Advancement in Research Trends (SMART)*, pages 251–255, Nov 2016.
- [6] Divya Sehgal and Ambuj Agarwal. Sentiment analysis of big data applications using twitter data with the help of hadoop framework. pages 251–255, 01 2016.
- [7] C. Sreejith, M. Indu, and P. C. R. Raj. N-gram based algorithm for distinguishing between hindi and sanskrit texts. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–4, July 2013.
- [8] V. U. Thompson, C. Panchev, and M. Oakes. Performance evaluation of similarity measures on similar and dissimilar text retrieval. In *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, volume 01, pages 577–584, Nov 2015.
- [9] F. Xu and V. Keelj. Collective sentiment mining of microblogs in 24-hour stock price movement prediction. In *2014 IEEE 16th Conference on Business Informatics*, volume 2, pages 60–67, July 2014.
- [10] R. Zhao and K. Mao. Fuzzy bag-of-words model for document representation. *IEEE Transactions on Fuzzy Systems*, 26(2):794–804, April 2018.