

Degree in Statistics

Title: A web scraping framework for stock price modelling using deep learning methods.

Author: Aleix Fibla Salgado

Advisor: Salvador Torra Porras

Department: Econometrics, Statistics and Spanish Economy

Academic year: 2018/2019

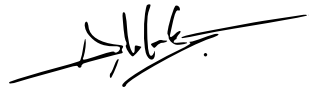


Declaration of Authorship

I, Aleix FIBLA SALGADO, declare that this thesis titled, "A web scraping framework for stock price modelling using deep learning methods." and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:



Date: 28/06/2019

Abstract

Aleix FIBLA SALGADO

A web scraping framework for stock price modelling using deep learning methods.

This work aims to shed light to the process of web scraping, emphasizing its importance in the new 'Big Data' era with an illustrative application of such methods in financial markets. The work essentially focuses on different scraping methodologies that can be used to obtain large quantities of heterogenous data in real time. Automatization of data extraction systems is one of the main objectives pursued in this work, immediately followed by the development of a framework for predictive modelling. applying neural networks and deep learning methods to the data obtained through web scraping. The goal pursued is to provide the reader with some remarkable notes on how these models work while allowing room for further research and improvements on the models presented.

Key words: Big data, neural networks, deep learning, web scraping, stock price modelling, time series.

AMS classification: 82C32 - Neural Nets.

Acknowledgements

Foremost, I should like to express my sincere gratitude to my advisor, Dr. Salvador Torra Porras. This work would not have been possible unless his continuous support, patience, motivation and tremendous knowledge in the field of neural networks and financial markets.

Besides my advisor, I also owe my deepest gratitude to Dra. Manuela Alcañiz Zanón, who encouraged me for initiating my studies in Statistics, and guided me through my entire academic career to become a statistician with her invaluable advice.

Last but not least, I express my deepest appreciation to Pau Casas Bonet and Silvia Sardà Rovira for assisting me with their expertise and insightful suggestions in financial matters.

Contents

Declaration of Authorship	i
Abstract	ii
Acknowledgements	iii
1 The Big Data era	1
1.1 Introduction to Big Data	1
1.2 A new programming paradigm	3
1.2.1 How do machines learn?	4
1.2.2 Machine Learning tasks	5
1.2.3 Training Data and test data	6
1.2.4 Assessing the outcome of learning	7
1.2.5 A primer in Deep Learning	8
1.3 Machine Learning in Finance	9
2 Web scraping Tools in Finance	11
2.1 The Net as a data source	11
2.2 Net interactions and networked programs	13
2.3 Overview of alternative data	14
3 Data Acquisition	16
3.1 Web Data Extraction	16
3.1.1 Getting data from yahoo finance	20
3.1.2 Getting news data from reuters	22
3.2 Variables	26
3.3 Exploratory Data Analysis	27
3.3.1 Company pricing data	27
3.3.2 News data	35
4 Methods	40
4.1 Theoretical considerations	41
4.1.1 Universal approximation theorem	41
4.1.2 Activation functions	42
4.1.3 Optimization	44
4.2 VADER Sentiment analysis	46
4.3 LSTM for time series prediction	50
5 Results	54
5.1 Unidimensional LSTM prediction	55
5.2 Multidimensional LSTM prediction	59
6 Concluding remarks	64

A Python class objects	66
A.1 Reuters news crawler	66
A.2 LSTM data processor	69
A.3 LSTM model definition	71
B Natural Language Processing	74
B.1 Stop Words	74
B.2 Punctuation List	84
B.3 Negate List	84
B.4 Booster Dictionary	85
B.5 Special case idioms	87
Bibliography	88

List of Figures

3.1	Example of DOM	18
3.2	Evolution of Adjusted Close Prices	28
3.3	Volume traded (in hundreds of millions)	29
3.4	Distribution of stock returns	30
3.5	Distribution of stock returns in 2018	33
3.6	SMA, 2012/01/01 - 2019/06/10	34
3.7	Stock returns cluster map	35
3.8	Adj close price cluster map	36
3.9	Candlestick chart MSFT	37
3.10	Candlestick chart JNJ	37
3.11	Candlestick chart JPM	38
3.12	Top words in headlines	39
3.13	Top words in news	39
4.1	Correlation between headlines and normal news sentiment scores	49
4.2	An unrolled RNN	51
4.3	LSTM architecture diagram	52
5.1	Unidimensional point-by-point prediction	56
5.2	Unidimensional multi-sequence prediction	58
5.3	Multidimensional multi-sequence prediction (compound)	60
5.4	Multidimensional multi-sequence prediction (volume)	61
5.5	Multi-sequence prediction on complete stock data	62
5.6	Full-sequence prediction on complete stock data	63

List of Tables

3.1	Stock data: Variables	26
3.2	Highest stock returns	31
3.3	Lowest stock returns	31
3.4	News from the day before abnormal returns	31
3.5	Stock returns standard deviation	32
3.6	Stock returns standard deviation in 2018	32
3.7	Number of news retrieved from Reuters	36
3.8	Average wordcount of headlines and news	37

List of Abbreviations

ACM	Association for Computing Machinery
AI	Artificial Intelligence
API	Application Programming Interface
BDA	Big Data Analytics
CSS	Cascading Style Sheets
DNA	Deoxyribonucleic acid
DOM	Document Object Model
EDA	Exploratory Data Analysis
GDP	Gross Domestic Product
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
ICA	Independent Component Analysis
IT	Information technology
LOOCV	Leave-one-out cross validation
LSTM	Long-short term memory
NLP	Natural Language Processing
NN	Neural Network, typically an Artificial Neural Network
PCA	Principal Component Analysis
SMA	Simple Moving Averages
SVM	Support Vector Machines
TCP/IP	Transmission Control Protocol / Internet Protocol
VADER	Valence Aware Dictionary and sEntiment Reasoner
W3C	World Wide Web Consortium

Chapter 1

The Big Data era

1.1 Introduction to Big Data

The last decade has seen the birth of what some call ‘the industrial revolution of data’. With advent of Web 2.0, the amount of published information and data collected is raising exponentially, up to the point that experts believe 90% of today’s data has been generated in merely the last two years¹. The aforementioned revolution encompasses, not only increments in the amount of data generated but also prominent advances in other drivers of the data process such as the pace at which data is coming in or the diversification of data sources.

The term ‘Big Data’ was first used in an ACM paper by Michael Cox and David Ellsworth in 1997 (Cox and Ellsworth, 1997). Despite this first reference, it was John Mashey, a year after, the one credited for coming up with the term Big data, endowing it with its modern meaning. These first voices noticed enormous increments in the amount of data that was generated year after year. By that time, the Big Data phenomenon was perceived as an incredible boost in the quantity and availability of potentially relevant data, sometimes referred to as an ‘information explosion’. Mr. Cox and Mr. Ellsworth defined the term ‘big data’ as follows: “Visualization provides an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of big data. When data sets do not fit in main memory (in core), or when they do not fit even on local disk, the most common solution is to acquire more resources.” A noticeable remark from their terminology is that big data was initially perceived as a challenge, in the form of a misalignment between data generation and data processing capabilities available by that time. Experts in the field alike (Denning, 1990; Lesk, 1997; Crane, 1997) were highly concerned about the real necessity of finding new ways of both data storage and processing, seeing a whole universe of opportunities in front of their eyes. The combination of this growing torrent of data and further advances in computing systems such as cloud computing, Hadoop or the rise of machine learning have largely contributed to the launch of the big data era.

Unfortunately, there is still huge controversy about the precise definition of Big Data, as it is a broad concept, related to many disciplines and encompassing different types of data. However, even though lacking a formal definition, it goes without saying that, in the last few years, Big Data has become a buzz word. The term gained popularity roughly across every field of study, creating applied disciplines of a wide variety of subjects. In general, the term Big Data is used to refer to any collection of data so massive and complex that exceeds the processing capabilities of traditional data management systems and techniques. Regarding the purposes of this paper,

¹ source IBM: Bringing Big Data to the enterprise.

(Jothimani, Shankar, and Yadav, 2018) provide an accurate description of what Big Data is. Authors argue that the term refers to large data being generated continuously in the form of unstructured data produced by heterogeneous groups of applications. The prior definition entitles three prominent characteristics, commonly used to give meaning to the term Big Data (see Laney, 2001).

Volume, Velocity and Variety, sometimes referred to as the three Vs, are the dimensions frequently used to characterize Big Data. Volume has probably the most straightforward connotation as it refers to the massive amount of data generated, collected and stored through records, transactions, files, tables, etc. The total number of bytes of data out there is truly mind-boggling and continually increasing at an accelerated pace². Experts have already coined the term 'astronomical scale' to describe the size of Big Data and there are numerous challenges related to dealing with this massive amount of information and the necessity to store it efficiently. Additional challenges arise in data processing due to volume, as more volume usually implies a higher cost and a worse performance.

Velocity indicates the speed at which data is flowing in or out. Contrary to traditional data collection techniques such as probability-based surveys, what we have now is a torrent of data flowing in continuously. It is even possible to say that nowadays data is "harvested" rather than explicitly collected, providing data scientists with real time data and automatic updates. However, data processing systems do not always match its production rate and opportunities are dismissed as sometimes it is extraordinarily challenging to deal with huge volumes of real time data.

Finally, Variety refers to the increase in the amount and heterogeneity of data sources. The time when only public institutions were responsible for dealing with data issues has come to an end; today trillions of data are generated from a wide variety of applications, devices, sensors, etc. Heterogeneity has usually a direct impact on the complexity of data generated. Even though tabular structured data is still predominant and relevant for data scientists, today a much wider variety of data structures are being used to solve problems. Images, text or geolocations are examples of more unstructured data which has created a world of opportunities but also made the analysis more complex to handle.

Recently, new Vs have been introduced to the Big Data community as new challenges and opportunities have been emerging. The first of these late Vs is Veracity. Big Data heterogeneity sometimes implies noisy, biased, uncertain or even volatile data. Veracity refers to the quality of the data and it is vital to make it operational as low quality data will inevitably lead to a poor analysis. Capitalizing on big data opportunities might offer a great advantage in decision making but, at the same time, data based decisions will be valuable only if data, the raw material of the whole process, is of a satisfactory quality and consistent over time. Reliability of the data source, accuracy, consistency of formats or relevance are aspects that definitely have to be taken into consideration when working with Big Data.

Together with Veracity, another V has recently come to the Big Data scenario and this one is Valence. Alike Valence in chemistry, a higher Valence means greater data connectivity. It measures the ratio of directly or indirectly connected data items that could occur within the connectivity potential of the collection. Holding this definition, it is clear that the Valence of a collection of data is dynamic and, apparently, it will increase over time enriching the information embedded and capturing data

² In the sixth edition of DOMO's report it is estimated that by 2020, every person on earth will generate 1.7MB of data per second.

from other networks. In a hypothetical universe of data, Valence might be seen as densities that measure the extent to which data is concentrated.

Nevertheless, what is at the heart of the Big Data challenge, is the capacity of turning all the previous Vs into truly useful business value. Conversations around Big Data take place everywhere whilst organizations keep on exploring new ways to effectively deploy these big volumes aiming to capture value for their stakeholders. The previously mentioned 'astronomical scale' has made experts skeptical about the prevalent imperative of always finding new approaches to deal with Volume. The focus now is shifting from size to value as for some 'big' is no longer the defining parameter, but rather how 'valuable' the data is. Accordingly, the term 'Smart Data' has come into play referring to feasible volumes that provide a reasonable number of insights.

In light of the above, it is possible to argue that Big Data has contributed to the rise of a data-driven era, where BDA are used in every sector and its possibilities appear to be endless. The growing of available data is a trend recognized worldwide and value is arising from this new resource. In fact, the list of opportunities that Big Data presents knows no boundaries. It can uncover hidden behavioural patterns and shed light on people's interests and intentions, in a way that makes even possible to anticipate events. Everyday, we all unconsciously leave behind our digital footprints and this data can yield extremely useful information. In the scientific sphere, BDA revealed the genetic origin of diseases, which ultimately led to personalized patient care, proved neutrinos have non-zero masses³ and even increased the understanding of our galaxy, bringing light to topics such as dark matter and dark energy. Regarding potential challenges or risks surrounding BDA, someone could think about how much data to store, how much this will cost, whether the data will be secure or how long it must be maintained (Michael and Miller, 2013). In addition, Big Data also presents new ethical challenges involving the preservation of privacy likewise needed to be addressed.

Overall, Big Data is changing our lives in both small and large ways. It delves deep into our understanding of fundamental issues and has already shifted our focus to experimenting rather than formulating hypothesis. Some may argue Big Data is even changing the scientific method as we have more data available and, consequently, testing becomes cheaper. This idea of experimenting and finding statistical evidence is closely related with the next section and, although improved powers of discernment and a better understanding of the real opportunities and risks involving Big Data are still needed, this phenomenon will unquestionably become an essential part of our lives.

1.2 A new programming paradigm

Troughout the last decade, Big Data has become a reality in more and more fields of research. However, it is also well-known that the ratio of data volume increase has surpassed the increase in processing power. In addition, data-collection technologies are becoming more and more efficient and advances like sensor networks, the Internet of Things, and others, foresee that this volume will continue to increase (Trifunovic et al., 2015). Today's major challenge for data scientists is to get real value from such quantities of data and, to achieve this goal, being capable of processing this huge amount of information is a must. In view of the limitations that traditional

³ This achievement came from the SDSS project, a major multi-spectral imaging and spectroscopic redshift survey using a dedicated 2.5-m wide-angle optical telescope.

processing units had when facing Big Data problems, added to the necessity of leveraging these cutting-edge resources, developers opted for shifting the programming paradigm to a more data oriented approach.

In general, programming consists in creating applications by breaking down requirements into composable problems that can be coded against. Before the Big Data era, developers used to figure out the rules behind the data, then write a code capable of making the calculations to finally output a result. However, with Big Data problems it is many times more reasonable to focus on the data rather than on the process, as both the bulk of data and its nature makes impossible to even think about any process behind it. This new paradigm consists in getting lots of examples of both inputs and outputs and use the machine to infer the rules between the data and the answers. Contrary to traditional programming, where rules and data were the inputs to the machine, Machine Learning rearranges this diagram by putting data and answers into the machine to get the rules out. Broadly speaking, Machine learning is all about feeding the machine with lots and lots of examples so that, regardless of the relationship behind the data and the answers, the computer is capable of learning such patterns and distinguish things. Its mathematical analogous would be finding a formula that maps X to Y .

1.2.1 How do machines learn?

Machine Learning originated in an environment where, although not at the same pace, the availability of data and computing power were rapidly evolving. These two drivers, fostered the development of new statistical methods and disciplines for dealing with large datasets as much of the information is valuable, only if there was a systematic way of making sense from it all.

Arthur Samuel, a pioneer in the study of artificial intelligence in 1950s and 1960s, said that machine learning is "the design and study of software artifacts that gives computers the ability to learn without being explicitly programmed". This initial grasp of what "learning" really meant to computers stressed the fact that, to be aligned with the volume of data flowing in, computers should learn from experience either with or without human supervision. A few years later, Tom M. Mitchell, a software engineer from Carnegie Mellon University, purposed a more formal definition: " A program can be said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E . Mitchell's rationale was based on the fact that learning must ultimately lead to better actions. Accordingly, computers can only be said to learn if they are capable of using experience to improve their performance on similar experiences in the future.

This conception of learning is remarkably similar to the human learning process. It starts with some data input (observation, recall, feel, memory storage etc.) followed by an abstraction process to translate the data into broader representations and finally generalize this abstraction to form a basis of action (Lantz, 2015). Notice these steps do follow the scientific method, with the difference that the machine know is responsible for the abstraction process which, in other words, implies formulating an hypothesis using events observed as the benchmark. The last stage involves testing the previously forged hypothesis to determine if it is stable and consistent enough to be generalized.

Likewise, The fundamental goal of machine learning relies on inducing unknown rules from examples of the rule's application to finally transform data into intelligent actions. In machine learning, data is no longer a set of X s to be fitted in some model,

but instead, it has to be treated in a more abstract way focusing on the information encoded in it. Furthermore, to build efficient algorithms it is also important to make an intelligent use of all the information present in the data and try to maximize its use because Big Data rapidly turns problems infeasible. As an example, think about a university exam in which the professor provides students with a set of ten thousand questions and tells them the exam will be fifty of them. Instead of memorizing ten thousand answers, it appears more reasonable to study a bunch of questions, get a general notion of the topics covered and then use this knowledge in the exam.

1.2.2 Machine Learning tasks

In line with the idea of learning presented in previous section, machine learning tasks and methods can be decomposed in two major types; supervised and unsupervised learning. They both profess the idea of utilizing experience to improve future performance but, at the same time, can be thought of as occupying opposite ends of the machine learning spectrum.

On the one hand, supervised learning is based on predicting an outcome using a ground truth as input or, in other words, having prior knowledge of the output values for the sample data. Supervised learning is usually adopted for problems where the program learns from a pair of labeled inputs and outputs, namely examples with the right answers. On the other hand, unsupervised learning aims to learn the inherent structure of the data without using explicitly labeled inputs. Since the ground truth is missing, it is sometimes difficult to assess the model's performance in unsupervised learning. However, these last machine learning systems might be highly useful in exploratory analysis, as they automatically identify structures in the data, and for dimensionality reduction. Some types of problems require semi-supervised learning methods, a collection of data and techniques located on the spectrum between supervised and unsupervised learning⁴.

By and large, the vast majority of supervised learning algorithms perform either classification or regression tasks. In both cases, the objective is to find structures or specific relationships in the input data that make possible to attain correct output results. Classification problems consist on predicting discrete values (labels, classes, categories) for one or more explanatory variables. On the contrary, in regression problems the program predicts a continuous value for the response variable. Examples of algorithms employed in supervised learning are SVM, Naive Bayes, Logistic Regression, Artificial Neural Networks and Random Forests, also named Decision Trees.

In the other side of the coin, clustering and dimensionality reduction are two of the most common tasks performed in unsupervised learning problems. Clustering assigns observation to different groups or *clusters* based on a similarity measure in a way that observations within a cluster are similar to each other but different from the ones in other groups. It is an extremely useful technique for exploring a dataset. Dimensionality reduction is another common unsupervised learning problem used when the number of explanatory variables is too large to work with. This method is based on detecting the explanatory variables that account for the greatest variability in the response and, consequently, remove from the analysis the ones capturing noise as their relevance will be lower. Among the most popular algorithms in unsupervised learning, we might encounter the wide range of clustering algorithms (see Aggarwal, 2014), PCA or ICA.

⁴See Zhu and Goldberg, 2009 for more information on semi-supervised learning techniques.

Depending on the user needs and the characteristics of the data available, one or other of these analysis is more advisable. In the case of labeled data, both supervised and unsupervised learning are plausible alternatives, whereas the last one is the only possible option in the absence of labels. Furthermore, supervised learning is generally used in predictive analytics while unsupervised learning can be employed in earlier stages of the data process for exploratory purposes like clustering, density estimation or representation learning. The majority of methods employed in this paper fall under the umbrella of supervised learning as "right answers" will be always available.

1.2.3 Training Data and test data

The collection of examples that comprise the learning experience in supervised learning is called the training set. Similarly, the set of examples used to assess the performance of the trained model is called the test set. Broadly speaking, the general procedure in machine learning is to fit the model on the training set, learn about data relationships and finally make predictions on data that was not trained. It seems reasonable not to include any training data in the test set because it will be difficult to evaluate whether the algorithm has learned to generalize from the training set or has simply memorized the outputs. The purpose of machine learning is to generalize relationships from the training dataset and infer them when dealing with new examples. A program that memorizes "too well" the training data by learning an overly complex model will output accurate predictions for the training set, but probably fail to predict response values for new examples. Memorizing the training set excessively accurately but not succeeding in test data predictions is called over-fitting, a problem that will be discussed in later sections.

Without regard to the algorithm used, training and test datasets are the two basic elements needed in Machine Learning and depending on the nature and structure of the data, both can be obtained in different ways. Most of the time, data is collected and stored in a single unit containing all the observations⁵ and the classic approach is to split this data into training and test datasets.

Theory does not provide specific requirements for the sizes of the partitions and they may vary depending on the data available and the speed at which the algorithm is capable of learning the underlying data structure. Indeed, there are a lot of factors influencing the previously mentioned speed rate like the algorithm's performance, the heterogeneity of the information or the amount of noise in the training dataset. As an example, think about an algorithm trained on a large collection of noisy, biased and sometimes incorrectly labeled data. Even with extra computing power and a long learning process, it is very likely the algorithm will not perform better than another one trained on a smaller but more representative dataset.

Some experts argue a third set of observations, called a validation or hold-out set, is sometimes required (Hackeling, 2017). Splitting the dataset into training and test data has its dangers. Although not a big concern in Big Data, the partition has to be random, in a way both subsets are small representations of the population. For instance, in classification problems all labels/classes/states must be present in both training and test data. The opposite will inevitably result in overfitting and cross validation is a good option to avoid it, particularly when training is scarce. Cross-validation is a deep topic, there are several methods and applications, also outside

⁵ each observation consists of an observed response variable and a set of observed explanatory variables.

Machine Learning (Arlot and Celisse, 2010). On the whole, the rationale behind cross-validation is closely related to bootstrap and resampling principles; estimates yield from different train/test splits should not be statistically significant assuming both objects are independent and observations are i.i.d. Therefore, it is a good way of verifying that the split is, indeed, random.

K-Folds and LOOCV are two popular methods used in cross-validation. The first one divides the training dataset in k partitions (or folds) and the algorithm is trained using all but one of them, which is left for testing purposes. Subsequent iterations rotate partitions so that, in the end, the algorithm is trained and assessed on all the training data. On the other hand, LOOCV treats each observation as one fold and builds the model using averages⁶. Leaving LOOCV aside, the number of folds considered represents a tradeoff between bias and variance errors, in addition to higher computational costs when more folds are added.

1.2.4 Assessing the outcome of learning

Recall, any learning process is characterized by turning abstracted knowledge obtained from inputs into a form that can be utilized for future action and improve performance on similar tasks. The term generalization is more than often used to refer to this idea of getting real knowledge out of observational inputs. However, in Big Data problems it is not feasible to examine one-by-one all the potential concepts behind the ocean of data flowing in thus, machines employ heuristics to make educated guesses about which are the most important concepts to be taken into consideration for further predictions. These heuristics necessarily generate imperfections in machine learning tasks, which are noticed when confronting the trained model with the validation set. The final step in machine learning problems is to determine the success of learning in spite of its prediction errors.

Unexplained variations in the data result in prediction errors which mainly fall under the umbrella of the model's bias and variance. The first one is associated with systematic sources of variability so that the model produces similar but biased results for an input regardless of the training set it was trained with. On the contrary, variance is more related to random variations derived from sampling the training data. Ideally, the objective would be to tackle both causes of prediction errors but usually decreasing one implies increasing the other. For this reason, this phenomenon is usually referred to as the bias-variance- trade-off and depending on the context, the optimal balance may vary. Models with high variance are inflexible but errors are easier to tackle while models with high variability often overfit the training data and much more difficult to generalize well to new cases. Solutions to overfitting are specific to different machine learning and most of the time rely upon intrinsic characteristics of the algorithm employed (Hawkins, 2004).

Identifying and understanding errors in prediction outputs is extremely important for model diagnosis and a first step through improvements. In the field of statistics, metrics are necessary since any decision should be supported with data facts and machine learning is not an exception. Metrics are extremely important to quantify the outcome of learning. In supervised learning, many performance measures are based on prediction errors whereas, unsupervised learning problems evaluate some attributes of the data structures discovered as error signals are missing. Finally, although somehow aligned, most performance measures are only valid for a

⁶ Notice how computationally expensive is LOOCV; the number of training sets resulting from all the combinations will equal the number of observations.

specific machine learning task and each problem should be evaluated using metrics that represent the costs associated with making errors in the real world.

1.2.5 A primer in Deep Learning

The term Deep Learning, refers to training neural networks, an artificial creation inspired by human neural networks. A regular NN consists of a bunch of processing units, usually named neurons, each one producing a sequence of valued activations of some process. These neurons can be activated with external stimulus or through connections with other active neurons, depending on the relative position of the neuron to the network and how connections are configured. Of course, this is how our brains work and throughout the last decade, the relationship between these neural networks and models used in mathematics and computational sciences have drawn the attention of a number experts in both fields.

By definition, Deep Learning is a sub-category of Machine Learning and they are both ought to fall inside the scope of AI. In its broadest definition, Machine Learning involves the design and implementation of algorithms than can learn from data. Deep learning is just one possible way of doing it, preaching the learning through examples and unveiling an outperformance in finiding patterns out of raw data.

Neural Networks are the heart of Deep Learning and they are nothing but approximations of functions arranged in multiple layers, which when multiplied by the input return an output close to the real value. As said before, a Neural Network is made of neurons, little nodes where the learning takes place. Each neuron takes a data input, computes a function and outputs new data. At the same time, these neurons are stacked together to compose the different layers.

The parallelism between human and artificial NN stems from (McCulloch and Pitts, 1943) usually regarded as the inception of NN in the theory of computation. The main idea presented in the article by McChulloch and Pitts (1943) is the creation of a logical apparatus to define the basic functioning of real neurons. The distinction between the input neurons and the rest, the two states of the neuron (firing and not firing) and predicates that can be computed by a given net; are concepts presented and developed in this early study⁷. A comtemporary picture of the scheme purposed by McChulloch and Pitts can be today's model of a neuron in a regular NN:

$$x_i[t] = \theta \left(b_i + \sum_{j \in \mathbf{C}(i)} w_{ij} x_j[t-1] \right) \quad (1.1)$$

At time t a neuron is in either a firing: $x_i[t] = 1$ or not firing state $x_i[t] = 0$. All synapses are charaterized and numerically quantified, depending on their strength, by a vector of weights: w which is positive for excitatory connections and negative for inhibitory connections. Notice the neuron also contains a bias parameter: b . This bias vector is nothing but units appended to the start/end of the input in each hidden layer to capture any constant needed in the model. A neuron is said to be activated when the sum of the previous weighted connections, that is from neurons j connected to it, plus the bias parameter is larger than zero. θ is the activation function or step funtion, and depending on the problem some are more efficient than others (Karlik and Olgac, 2011). The last element: $\mathbf{C}(i)$ is the set of neurons that

⁷ Ideas developed in the paper by McChulloch and Pitts (1943) are highly complex and hard to follow nowadays. (Kleene, 1956) purposes a much more clear explanation of the same concepts.

impinge on neuron i . Accordingly, the state of neuron i will be determined through the weighted addition of values from previous neurons in C .

Arranging multiple processing layers with different activations allows computational models to learn representations of data with multiple levels of abstraction. Starting with the raw data, each layer transforms previous level representation into a new one at a higher, more abstract level. With enough transformations, very complex data patterns can be learned and irrelevant information is discriminated at each level. A deep-learning architecture is a multilayer stack of different modules, all (or most) of them subject to learning and many of which compute non-linear input-output mappings (LeCun, Bengio, and Hinton, 2015). These mappings are driven by a loss function that measures the error between the hypothetical layer output scores when compared to the real values. Each layer adjusts the weights by means of a gradient vector which indicates the marginal increment or decrement in the loss function if the weights were increased by a small amount. Finally, before moving on to the next layer, the weights are modified in the opposite direction to the gradient vector and this output will be the input to the new module. Repeating this procedure a number of times allows the machine to obtain a very precise X-Y mapping which can be applied to new input data and continue with the learning process.

The major success of Deep Learning is that layers of features are not designed by humans, contrary to classical inference. Machines are empowered with total freedom to determine the optimal mapping from whatever function of data and weights using a general-purpose learning procedure. The field is making major advances in solving previously unfeasible problems with enormous contributions in many domains. Image and speech recognition might be the most popular applications of Deep Learning techniques but do not forget, predictive modelling in DNA coding and gene expression, particle acceleration or natural language understanding using methods such as sentiment analysis or topic classification (LeCun, Bengio, and Hinton, 2015).

In the near future, Deep Learning is expected to have even a greater success because it requires very little human intervention and due to its high flexibility and machine oriented approach it can be applied to almost any discipline and can take advantage of the tremendous increase in the amount of data available in this new Big Data era. New architectures and algorithms are continually being developed and machines have proven to become better year after year at prediction tasks thanks to Deep Learning, a fact that will undoubtedly accelerate its expansion.

1.3 Machine Learning in Finance

Economic and financial prediction models are of great relevance and general interest. Just imagine the possibilities that might offer being able to anticipate an increase in the price of a commodity, foresee GDP growth rate for next semester or predict future returns from a company's shares. Having read previous sections, someone might think that Machine Learning is the key to succeed in all these predictive tasks but, without being false, the financial world is sometimes quite daunting. Economic theory is more than often unclear about the variables that influence a certain output and it suggests that many of them are spread throughout different types of data for instance economic, social or political. Dealing with such disparate data sources is not easy and if researchers pursue to capture all factors that might impact some financial magnitude, the collection of potentially relevant data will probably become too large to handle. In addition, the way these sources interact is extremely complex,

presumably non-linear functions, which are not properly specified by the theory. As a consequence, a number of predictive models in finance are built via trial and error with little theoretical justification and sometimes accuracy is adversely affected.

In the world of finance, artificial intelligence could be helpful in a lot of different areas and the value of machine learning is becoming more apparent by the day. Over the past few years, the big players in the industry have witnessed profound changes with the entry of new joiners adopting quantitative investment methods to compete in the marketplace. The so-called Fintech companies have experienced great success in bringing financial services to a new level where machine learning plays a key role. In response, big firms had to adopt an AI approach in some of their services, because if not they would have been left behind in not many years. The flow of information in financial markets is now quicker than ever and machine learning seems to be the key for investors to chase these fleeting opportunities and keep up with a highly sensitive environment.

The applications of machine in the financial sector are huge and diverse. Both regression and classification methods are extremely useful in areas such as fraud prevention, risk management, investment prediction, customer service, loan underwriting, document interpretation or algorithmic trading; among others. This paper focuses on the uses and functionalities that big data and machine learning can bring to stock prediction with emphasis on its advantages against traditional analysis methods.

Stock prices are ought to be very volatile, dynamic and susceptible to quick changes as they are supposedly influenced by multiple factors both within the financial domain of companies and outside the scope of financial markets. With proper information about stock prices with a suitable level of accuracy, the overall loss on investments might be reduced and the whole society can benefit from it through a better allocation of resources, avoiding wastes and promoting a healthy expansion of financial markets where people can invest more confidently. Transparency is definitely an issue in investment for all the stakeholders involved in the process and machine learning provides a framework where truthful informed financial decisions are about to be a reality. However, as said before, lots of unknown factors surround stock prices and the expertise from professional trades is hard to replace. In addition, famous theorems like the efficient market hypothesis or the random walk hypothesis come to say that future stock prices are completely unpredictable and they do not depend on past data, henceforth there are not patterns to be exploited since trends do not exist (Shah, 2007). Machine learning and Deep Learning methods are now challenging these theorems and it might be still early but investors willing to adapt these state-of-the-art technologies will likely have an edge in the new financial sector.

Following sections introduce a stock price forecasting framework where NN are used to predict future stock movements and alternative data is also feeded into the model in an attempt to capture potentially relevant unknown factors and enhance the accuracy of the model, ultimately seeking to avoid blind investment behaviours. Webscraping methods are used in data acquisition stages since their vision is perfectly aligned with the Big Data train of thought posed in this chapter. Next section presents a detailed outline of webscraping techniques together with the reason why automatizing data acquisition and even pre-processing is necessary to exploit all the opportunities that stem from Big Data.

Chapter 2

Web scraping Tools in Finance

2.1 The Net as a data source

It is a well-known fact that the World Wide Web is immense and growing at an extraordinary fast pace every second. Its exact size, however, is difficult to determine as it is hard for a counter to tackle small IT sites and a large portion of the Web, popularly known as the Deep Web, is hidden and not easy to access. Without forgetting the above considerations what we can say is that, in terms of weight, the World Wide Web is measured in zettabytes (1 zettabyte = 10^{21} bytes) and so forth, such a gigantic universe of data presents a major opportunity to turn it into valuable insights. The scope and potential of the abundance of data in the network sounds exciting but it can be both blessing and curse. As probably the biggest source of information available in the world, the World Wide Web is an optimal illustration of the Big Data framework discussed in previous chapter. Lots of relevant information, as hidden treasures in a vast sea of data; along with variety, velocity and veracity issues.

The data growth rate on the internet is soaring every year and, as a consequence, data comes predominantly in an unstructured form (Chaulagain et al., 2017). Since this data is not prepared neither to be fitted in any model nor to be analyzed in any way, information churning might become a problem. Furthermore, the data on the web is a constant state of flux; it is continuously updated and modified. Therefore, web-based data collection techniques must be fast and flexible enough to keep pace with this constant change. Ultimately, due to the voluntary and often anonymous nature of user interactions with the Web, added to the lack of a body responsible for assessing the quality and availability of the data, some of this information is always surrounded with uncertainty. In light of the above, and with the irruption of Big Data technologies, it is becoming more and more crucial for both researchers and practitioners to find efficient ways of exploring the network and retrieving relevant information. Harnessing such volumes of heterogenous data from the web often requires a highly customizable programmatic approach (Krotov and Tennyson, 2018a) although, for those not familiar with programming languages, today's web is becoming overwhelmed with handy resources that most of the time will do the work for you.

Data acquisition is the starting point of any data science project and the inputs can be obtained from either private sources like company's internal data or public sources, and it is here where a world of possibilities opens up. Public sources comprise official data from government institutions, journals or, more generally, any data that can be found on the web regardless of whether it is open data or you have to purchase it. All the data employed in this paper falls under the umbrella of open data and it is obtained from the net through webscraping. At odds with what most people think, open data not only refers to tidy and structured data published on the web by some entity. The concept is more general and, in fact, it does not conflict

with how data is structured, presented or shared. Open data merely refers to the belief that some data should be freely available to everybody and all players should be able to use and republish it without any copyright restrictions, patents or other mechanisms of control (Berners-Lee, Hendler, and Lassila, 2001). Therefore, no matter how the data looks like or where it comes from, every byte of information free to use, reuse and redistribute is considered open data.

In addition, this paper seeks to exemplify previous arguments regarding the World Wide Web and reinforce the idea that the network is vast and messy but, with the proper tools, is possible to extract real value out of this universe of information. As said before, there are different ways of gathering data from the web; some users may interact with an API, others may use data mining or text mining and the ones lacking programming skills will simply click on the download link of a data provider website. Throughout the last decade, different terms have been coined to refer to this act of extracting data from websites: web scraping, web harvesting, web crawling or web data extraction; are just a few examples. However, the definition of this terminology is still vague and more than often leads to confusions and controversies. As an example, for many authors alike (Krotov and Tennyson, 2018a, Boeing and Waddell, 2017), although the term typically refers to automated processes, web scraping is solely the act of extracting data from websites. Accordingly, manually downloading data from a website could be considered web scraping in the most user-friendly level. Others understand web scraping from a standpoint close to parsing text files formatted in HTML and XPath (Krotov and Tennyson, 2018b). In theory, web scraping is the practice of gathering data through any means other than a program interacting with an API or a human using a browser. A text mining approach does not seem wrongheaded as the previous interpretation is typically accomplished by a program that queries a web server, requests data (usually html) and parses the code obtained to extract the useful information. However, recent studies (Lawson, 2015) put the emphasis on the web exploration step arguing that the real added value that webscraping offers is the possibility of gathering and processing large amounts of data rather than viewing one page at a time which is what a browser can attain. Therefore, with convenient scrapers, one can access databases spanning thousands of webpages at once. Again, this can be attained through different methods and one more time text mining is an option since changing dates in a web link is usually enough to access historical data. In the absence of consensus regarding the boundaries of webscraping and in line with the purposes of this work, any retrieval of information from the web entailing some sort of automatization together with an organized approach will be considered webscraping.

Likewise analysis where the data is obtained in the form of a table from an official data provider website or by querying a database, here the source is the net itself and the goal pursued is to find efficient ways of exploring it, downloading the desired information and clean it so the final output from this initial data acquisition step is a tidy dataset ready for further modelling. This objective is achieved through web-scraping and in this respect, it involves not only retrieving the data from the web but also, when required, turn unstructured data into a tidy and properly arranged dataset. Finally, as said before, the whole process should be fully automatized and reproducible capturing updates and outputting real time results.

2.2 Net interactions and networked programs

Contrary to what most people think, the World Wide Web and the Internet are not synonyms. The first, refers to a decentralized interconnection of different devices by means of a set of protocols known as TCP/IP protocols, while the World Wide Web is a system that manages the information shared over the Internet through the so-called HyperText Transport Protocol or HTTP. This network protocol is actually quite simple and there are multiple built-in supports in almost all programming languages that make it relatively easy to create network connections and retrieve data over them. These modules are like files, except that they provide a two-way connection between the network and the device in which they are executed. Therefore, both sides can read and write to the same program and the protocol sets the rules that regulate this user network interaction ¹.

In today's world, almost every person in developed countries navigates the network everyday but, for the vast majority, its mechanisms still seem quite mysterious. Computer interfaces have progressed to the point where the net can be extremely valuable for users even though they do not have any idea about how it works. Internet browsers are the main responsables of this and, although it is not necessary to understand what the network is doing in all our interactions to succeed at web-scraping, web data extraction techniques usually require to connect to the network at a lower level. In fact, the exchange of information between a user and the network takes place without the intervention of a browser. Broadly speaking, the Internet is only the mean to request some action from a web server and the browser is merely code that interprets the server response and gets the user back the same data in more attractive formats such as images, videos, music, formatted text etc ². As said before, capabilities of web browsers are also present in almost all programming languages and the value that own-built webscrapers can bring to the analysis is the opportunity to interpret server responses in the most convenient way.

The main output of the previously mentioned request/response mechanism regulated by the HTTP protocol is an hypertext document, commonly known as a webpage. HTML is the predominant language for encoding web architectures and it is largely supported by W3C consortium. HTML pages can be thought of a form of semi-structured data following a nested structure where information is stored and accessed using tags. Each HTML element is associated with a specific tag, represented with angled brackets, and typically coming in pairs for both opening and closing the element. These tags can be also thought as element insertions in a webpage they can be profitably used in the design of suitable scrapers. The list of elements in a webpage and their corresponding tags is lengthy, including hyperlinks, images, paragraphs or videos, among others. However, although most of the webpage content is found in the HTML code, it is also important to acknowledge that webpages might also include other items such as CSS or Javascript objects. These extra features are not particularly relevant for the purposes of this paper but definitely a factor to be taken into consideration from a programmatic perspective since languages are different and they can be misleading.

¹ See a description of the HyperText Transport Protocol in the following document: [Hypertext Transfer Protocol – HTTP/1.1](#)

² Internet browsers are relatively new compared to the Internet, first ones were released in the nineties, and, as any other piece of code, they can be re-written, re-used or re-formated (Lawson, 2015).

Literature provides a number of different strategies and techniques adopted in the field of Web Data Extraction (Ferrara et al., 2014) and most of the time human expertise is required to define the rules of the data extraction process. The challenge is to establish a high degree of automatization to reduce human efforts as much as possible and this often goes hand in hand with having a good knowledge of the domain. (Krotov and Tennyson, 2018b) propose a general three-step process to characterize web-data extraction strategies:

- The initial phase involves examining the website architecture using a web inspector to understand how the data is stored at a technical level. The objective is to get a general idea of how the information is encoded, likely HTML language, and feed the scraper with these inputs. Ideally, the underlying structure of the website captured by the scraper should be extrapolated to others of similar nature.
- Second phase concerns running a previously developed piece of code that automatically browses the web and retrieves the desired information. The understanding of the domain comes to play in this stage as it will guide the scraper to fetch the needed data.
- Finally, there is also a "Data Organization" step which involves cleaning the data retrieved, sometimes coming in an unstructured form, and arrange it in a dataset. This last stage makes the data ready for further analysis.

Even though the previous sequence is coherent, the three stages are often intertwined. Many times, the programmer will have to go back and forward until the final tidy dataset is obtained since the process might have to be applied, for instance, to smaller data units like rows or individual chunks of information retrieved from different websites. One more time, the common note along all stages is human supervision, particularly when analyzing the website architecture. It is probably the most tedious part and where more inefficiencies are concentrated since some domains often require to be examined individually.

Over the last decade, various strategies have been developed in the field of web scraping to deal with previously mentioned concerns and reduce the commitment of human domain experts. A possible way forward involves the adoption of Machine Learning algorithms capable of identifying the structure of webpages and extracting the useful information. Some of these systems are discussed in further research sections but the basic outline consists of training a program with numerous examples of web architectures so that after the training it becomes competent in autonomously extracting data from similar, or even different, domains.

2.3 Overview of alternative data

In general, and particularly in the world of finance, the idea of bringing to the stage all the Big Data culture is to empower businesses with new insights and foster the development of data driven models for decision making to ultimately create value. Recent advances in web data extraction techniques enable a more efficient search and retrieval of information from different sources and allow traditional datasets to be enriched with new data. In line with previous discussion about the factors that might have an impact on some financial indicator, more than often traditional data used in finance or economics fail to capture enough inputs to conduct accurate forecasts [1](#). One of the contributions of Big Data in this field involves the possibility of

enriching traditional datasets like production rates, historical financial results from firms or GDP series with other types of data that will hopefully seize external variables that might impact a certain output but have not been contemplated before. The informational advantage provided by these new sources of data can be in the form of uncovering new relevant information not included in traditional datasets, or anticipating it before traditional sources do. This new information is popularly known as alternative data and, although veracity issues are not the major concern as many times the model itself will deploy inaccurate information, the three Vs are more than valid to characterize these new sources. The term alternative data is chiefly used in finance with little visibility in other fields of research and it refers to all the information gathered from non-traditional data sources which can provide new insights beyond those which traditional data is capable of providing.

The concreteness of what does alternative data include varies from industry to industry. In finance, the market of alternative data is quite fragmented but recently, JP Morgan developed a framework to classify alternative data into three main categories based on the manner in which data was generated:

- **Individuals:** Information collected at an individual level that will provide insights about market trends, consumer behaviour, public opinion, etc.
- **Business Processes:** Data from agents in the market with a new higher level of detail.
- **Sensors:** This category includes data from satellites or geolocations.

More often than not, it is not easy to evaluate the relevance that a specific type of alternative data might have on a financial magnitude. In addition, there is little standardization for most alternative data offerings whereby most of the time the data will come in an unstructured or semi-structured form. Accordingly, pre-processing is usually a must before getting this data ready and compatible with traditional datasets. For this reason, webscraping is usually the mean to gather alternative data since a programmatic approach and an automatized pre-processing is required to make the whole process feasible.

There are lots of success stories related to the use of alternative data sources to improve financial processes, specially in predictive modelling matters. A few examples are the use of consumer transaction data to trade individual stocks, twitter sentiment data to trade the broad equity market or geolocation data to estimate real activity in some area. In this paper, the power of alternative data is put to test using sentiment data from news' headlines to make stock price predictions more accurate. The idea is that news data will provide the model with insights about market trends and expectations so that the program will learn how to anticipate future changes and outscore stock markets in reaction time.

Chapter 3

Data Acquisition

First step in any data science project involves the acquisition and pre-processing of the data required for further analysis and modelling. In line with the idea developed in previous sections, the underlying principle behind the obtention of the data is strongly based on webscraping, attempting to reduce human efforts as much as possible, and offering a user-friendly approach with a higher degree of automatization. These procedures will most likely be reproducible with a 'manually' data obtention process but, and this is one of the main justifications for using webscraping, in a context as dynamic as financial markets, velocity of data is a success factor and the optimization of processes is essential. Furthermore, when deciding which data to use in a project its cost is an important consideration. As said before, all the data employed is open data hence there are not any direct costs associated with its purchase. Nevertheless, the opportunity cost of time invested in acquiring and analyzing the data is also relevant and webscraping pursues to generate savings here.

The Big Data universe is changing the foundations of the entire data science process and state-of-the-art methods are a valid patch when resources are scarce or simply when the researcher does not have any theoretical input behind the phenomenon studied. Today's world requires more flexible, adaptative and faster methods to deal with problems that most of the time demand a prompt response and theoretical considerations are limited. Machine Learning, deep learning and all these new analysis tools might be the most widely recognized exponent of old methods in data science being replaced with modern techniques. However, not only the analysis step has suffered changes, innovations in data acquisition or preprocessing are also remarkable, largely due to webscraping. In the not too distant future, data storage will be a serious concern for computer manufacturers and the current trend is in the direction of web resources and interconnected servers, trying to minimize the amount of information locally stored. Therefore, in data extraction, a webscraper, designed ad-hoc for a specific problem, seems to be the most efficient way to obtain the desired information while optimizing the capacity of memory resources.

3.1 Web Data Extraction

The majority of approaches to extracting data from the Web are designed ad-hoc to operate on a specific domain¹. Accordingly and, although most of the time programming languages provide users with built-in modules that ease the work, certain programming skills and a good knowledge of html language and data structures are desirable to built efficient scrapers.

¹ (Ferrara et al., 2014) discusses the potential of cross-fertilization methods and the possibility of re-using Web Data Extraction techniques originally developed in a given application to multiple domains.

Programming languages like Python or R have utilities called sockets which are nothing more than regular files with the peculiarity that they provide a two-way connection between two programs thus the user can both read and write to the same socket. These sockets use the HTTP protocol to emulate a browser and make network connections and data retrieval very easy to handle.

```
1 import socket
2
3 mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4 mysock.connect(('data.pr4e.org', 80))
5 cmd = 'GET http://data.pr4e.org HTTP/1.0\r\n\r\n'.encode()
6 mysock.send(cmd)
7
8 while True:
9     data = mysock.recv(1024)
10    if (len(data) < 1):
11        break
12    print(data.decode())
13
14 mysock.close()
```

LISTING (3.1) A simple socket

This piece of code makes a connection to a web server and follows the HTTP protocol rules to request some information and display it on python command line. Once the program sends a get request followed by a blank line, it is told to receive data in 1024-character chunks from the socket and print it to the screen until there is nothing more to read (recv() returns an empty string). Notice the output is basically html code enriched with headers' information. The same can be done by opening the url in a web browser with a web developer console and manually examine the page source code. Html webpage code includes several tags, as discussed in chapter 2, which also offer the possibility to access new related webpages and, consequently, more information ².

Sockets can be thought of as low-level network interfaces used to communicate with a web, mail or many other types of servers. In Python, the coding approach is highly object-oriented thus the socket() command returns a socket object with a number of methods that allow programmers to connect, read, write or send information from and to the network. The functioning is considerably simple, the only thing necessary is a written protocol describing which rules to follow while the interaction takes place. However, since most of the time the protocol used is HTTP, programming languages have already special libraries that support HTTP protocol, such as urllib in python or urltools in R, for information retrieval over the net. Notice in previous example the code read through plain text but same approach can be considered when retrieving images or music files.

Urllib in python ease the process of getting information from the net by treating webpages as files. It is only necessary to indicate which domain to retrieve and the library will figure out how to deal with HTTP protocol and header issues.

```
1 import urllib
2
3 fhand = urllib.request.urlopen('http://www.ub.edu/monub/')
4 for line in fhand:
5     print(line.strip())
```

LISTING (3.2) Webpage retrieval

² This can be done, for instance, by appending *intro-short.txt* to the GET request

The output object is the html contents of the webpage and it can be treated as sort of a text file (although still byte literal) and, accordingly, read through it using a for loop. This level of interaction is higher than using a socket as the information is ready to be deployed and, up to this point, it is even possible to perform simple tasks such as word counts or tackle specific tags that might provide the desired information. As an example, differences aside, the google search engine works at this level of abstraction; it looks at the source code of one webpage, extract links to other related pages, retrieve them, and finally use its famous page rank algorithm to display the most relevant finds in response to a user search. Web spiders use this principle to navigate the web, as it quite straightforward since links within a webpage are defined with an 'a' tag in html thus not even complicated regular expressions are needed to find them ³.

The semi-structured nature of webpages, which is so characteristic of html code, is one of the most exploited features in web data extraction. Such data structure, can be naturally represented as a labeled ordered rooted tree, where tags illustrate the different nested levels of the webpage source code. The representation of a webpage in this form is referred to as DOM which is basically a way of converting html webpages to plain text containing the tags and particular keywords that can be easily interpreted and accessed. For further details on DOM trees consider the following link: www.w3.org/DOM.

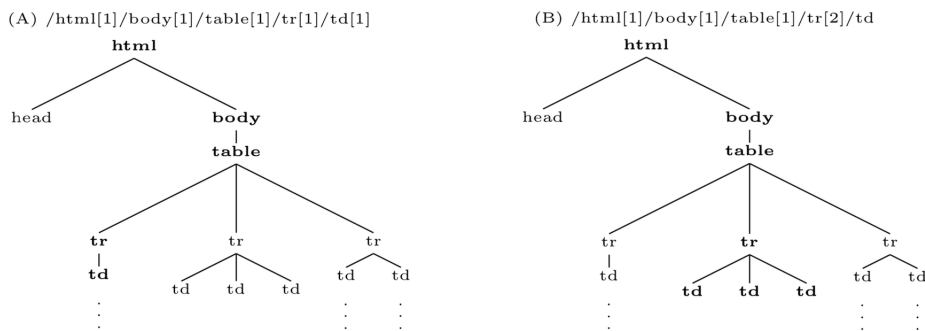


FIGURE (3.1) Example of DOM

Source: (Ferrara et al., 2014)

The DOM tree structure illustrated in figure 3.1 is, in fact, the output of listing 3.2. HTML code might look appealing to some and they are not mistaken, the information is there and ready for further steps where lots of interesting things, like the previously mentioned spider webs, can be done. HTML can be parsed using regular expressions to repeatedly search and extract strings that match a particular pattern like links, data within tables, images, headers, etc. Indeed, the adoption of DOM objects for html language offers the possibility of exploiting features inherent to XML languages since both are of similar nature. Converting webpages to some XML form is certainly something to be taken into consideration since regular expressions work nicely when html is predictable and properly formatted but a solution based on only this approach might miss some valid information or completely fail to retrieve anything when dealing with broken website content.

In order to solve the broken content issue and provide web scrapers with extra robustness, several libraries have been developed in many different programming languages and environments to help users parsing html code. In Python, the most

³ Google spiders its way through nearly all the existing webpages in the net.

relevant and widely used is *BeautifulSoup*, a library built to work with data in either html or XML formats. Amongst other capabilities, *BeautifulSoup* provides ways of navigating, fetching and transforming the parse tree. Its main functionality lies in the fact that the gap between HTML and XML is not always savable. At times, html is broken in ways that will cause an XML parser to reject the entire website and, in this respect, *BeautifulSoup* tolerates flawed html and still lets the user extract the data easily. JSON is another plausible option for handling web data although it has fewer capabilities than XML. Conversely, it has the advantage of simplicity since it is not barely more than a combination of Python dictionaries and lists and, accordingly, the mapping is much easier. Bottomline, data exchanged between applications through the net is certainly complex and, to some extent, difficult to work with. In consequence, the tendency is to use XML, JSON or other nested data structures to represent this data, improve usability and ease the exchange of information between cooperating programs.

The process of turning raw html from a particular website into a handy text file in the form of XML, JSON or, even better, a python dictionary can be seen as the highest level of interaction with the net since the user is giving a web adress as an input and the program outputs a text object with the desired information nicely formatted and ready to be analyzed. Nonetheless, the human effort is still notable here as all the steps have to be explicitly written by an expert and, as discussed earlier, the vast majority of designs are ad-hoc. In addition, it can also happen that the depth of data required is outside the boundaries of what a regular machine can reach and efficiently collect. For these reasons, big tech companies like Google, Amazon or Oracle have developing a number of databases containing tones of bytes of data impossible to be collected by scanning the net with limited resources.

The next step to webscraping concerns the use of Application Program Interfaces or APIs to benefit from a particular service offered by a website application. The techniques used are very similar to the ones described above but usually more automatic. An API can be seen as an application-to-application contract in which a program provides a set of services to those who request them under certain published rules. Traditionally, computing has been based on a single individual device connected to the real world by a set of inputs and outputs. The big data era and, particularly, the new IoT brought to stage higher complexity of tasks that originated deployment difficulties and serious test and maintenance issues to those individual applications (Johnston and Burnett, 2012). Perhaps, it is more efficient to include in the functionality of our code access to services provided by other programs, in different IoT network environments, in order to meet current programming needs. This last approach appears to be the tendency nowadays where interconnection of devices and shared network environments seem to prevail over single standalone applications.

Service-oriented architectures or SOA is the term commonly used to refer to the abovementioned programming approach where one application makes use of the services of other applications. This type of design has many advantages, including that the owner always maintains a copy of the data and also has the right of setting the rules involving the use of their data. APIs are highly diverse and can take different forms but, if properly designed, they aim to act as building blocks to help developers in creating new programs. In developing new applications, APIs simplify the programming stage since they abstract the underlying tasks and expose only the options or actions required by the user. Furthermore, an API can provide critical support in setting up a web-based system, operating system or a database; among others. There is a special nomenclature for those applications offering services via

their APIs over the web, they are popularly known as web services.

Data acquisition by means of one or many APIs is probably the highest level of interaction with the network since most of underlying implementations, which were clearly visible and explicitly programmed in the socket case, are no longer needed to be set but, instead, driven by the same API. All the previous net interactions, with their respective abstraction levels, ought to fall within the sphere of webscraping although, as seen, the way the code is written is totally different.

Subsequent sections and subsections present the two types of data used in further models. In both cases, the major bulk of work is conducted by different APIs despite basic knowledge about network interaction and html language will be needed at some point. The core data for the experiment is mainly stock data retrieved from yahoo finance by means of the pandas-datareader, a Python library that offers up to date remote data access for pandas, another open source library (BSD licensed) used for handling data structures and basic analysis in Python. If familiar with pandas data structures, the pandas-datareader is an optimal option since it extracts data from different Internet domains such as Google Finance, Enigma, Morningstar, Eurostat or the World Bank ⁴ via their respective APIs, and turns it into a pandas dataframe. Similarly, a news scraper is designed to get data from Reuters, an international news agency with a web portal very suitable for scraping purposes.

The abovementioned data sets are the raw materials needed in the analysis going on in this paper. The first and core, a collection of financial data from stock listed firms. Second, an assortment of news that will enrich the information provided by financial markets. Complementarily, webscraping offers a systematic approach to the obtention of data and object-oriented programming, so characteristic of Python, will be present in both scenarios.

3.1.1 Getting data from yahoo finance

Yahoo finance is a media property belonging to Yahoo! which provides financial data, including stock quotes, press, releases, financial reports and also technical indicators. As of this writing, Yahoo Finance's API is working perfectly and within the scope of pandas-datareader library, although it is not unusual that some API cease their activity and, if that is the case, it will be unavoidable to resort to other sources like Google Finance or Morningstar.

The focus will be on S&P500, one of the most important indexes in the world containing the 500 largest American firms (in terms of market capitalization) listed on the NYSE, NASDAQ or the Cboe BZX Exchange. The justification behind using S&P500 is because data of big companies is always easy to obtain and less likely to include errors or inconsistencies. Furthermore, these firms are more relevant in global press and the probability of getting a sample of news larger and representative enough is higher than if working with smaller companies.

Pandas-datareader uses the ticker symbol of a stock to fetch the stock's data from yahoo finance. Tickers are stock codes used to uniquely identify public traded shares of a particular stock in a particular market. Accordingly, the first input to obtain is the desired tickers to pass them as arguments in pandas-datareader functions. Even though is possible to look for the ticker symbols of S&P500 firms, copy paste them in a plain text file and then read the file in Python, the list of S&P500 firms is posted in several webpages and, in line with the idea of reducing human efforts as much as possible, a webscraping approach is more convenient.

⁴ Stable documentation is available on github.io.

```

1 import requests
2 import bs4 as bs
3 import pickle
4
5 def save_sp500_tickers():
6     resp = requests.get('http://en.wikipedia.org/wiki/List_of_S%26
7     P_500_companies')
8     soup = bs.BeautifulSoup(resp.text, 'lxml')
9     table = soup.find('table', {'class': 'wikitable sortable'})
10    tickers = []
11    for row in table.findAll('tr')[1:]:
12        ticker = row.findAll('td')[0].text
13        tickers.append(ticker)
14
15    with open("sp500tickers.pickle", "wb") as f:
16        pickle.dump(tickers, f)
17
18    return tickers

```

LISTING (3.3) S&P500 Tickers

The Python function defined in listing 3.3 requests the source code of a wikipedia webpage [List of S&P500 companies](#) containing the tickers for companies in the S&P500 index. Wikipedia is just an option, it is highly likely to find the tickers needed in other webpages. After requesting the primary information, BeautifulSoup is used to parse the html code creating a "soup" object. Next commands tell the function to get the first column of the table data lines within the wikitable sortable object. Ultimately, the list is locally saved with the pickle module, which automatically serializes Python objects to a byte stream.

Next step involves the website crawling stage. The previously obtained list of tickers, after de-serialized, is passed as an argument to pandas-datareader to hit the yahoo finance API and obtain stock data.

```

1 import datetime as dt
2 import time
3 import pandas_datareader.data as web
4 import os
5 import pickle
6
7 def get_data_from_yahoo(reload_sp500=False):
8     if reload_sp500:
9         tickers = save_sp500_tickers()
10        tickers = [ticker.replace('.', '-') for ticker in tickers]
11        tickers = [ticker.rstrip() for ticker in tickers]
12    else:
13        with open("sp500tickers.pickle", "rb") as f:
14            tickers = pickle.load(f)
15            tickers = [ticker.replace('.', '-') for ticker in tickers]
16            tickers = [ticker.rstrip() for ticker in tickers]
17    if not os.path.exists('stock_dfs'):
18        os.makedirs('stock_dfs')
19
20    start = dt.datetime(2010, 1, 1)
21    end = dt.datetime.now()
22    for ticker in tickers:
23        # save progress just in case connection breaks
24        if not os.path.exists('stock_dfs/{}.csv'.format(ticker)):
25            df = web.DataReader(ticker, 'yahoo', start, end)
26            print(df.head())
27            df.reset_index(inplace=True)
28            df.set_index("Date", inplace=True)
29            df.to_csv('stock_dfs/{}.csv'.format(ticker))

```

```
30     else :  
31         print('Already have {}'.format(ticker))  
32         time.sleep(0.5)
```

LISTING (3.4) S&P500 Stock data

Notice listing 3.4 calls the previously defined `save_sp500_tickers()` inside the flow of execution. Another important note is that BeautifulSoup is no longer used here since we are not retrieving any html code but, instead, the API is responsible for the parsing issues. Once retrieved, stock data is stored in a previously created directory named `'stock_dfs'` incorporating the information of all companies in the ticker list. Individual files are stored in csv format but any other tabular format is also advisable. A time horizon from 01/01/2010 until the current day has been settled although the real data range employed will vary depending on computing resources available since, and next section will evince that, obtaining news data to complement stock prices is extremely costly⁵. Additionally, a control statement is included to save progress in case connection breaks.

Around 30 lines of code are enough to build a systematic approach for downloading all stock quotes from S&P500 companies and the execution is fully automatic. Nevertheless, due to computing power limitations and in line with previous argument of targeting firms with numerous and rich news data, only three companies in S&P500 have been considered for further analysis and modelling. Firms selected are sector leaders in terms of relative weight with respect to total US stock market capitalization. Another idea behind this filtering is to end up with companies belonging to different sectors to analyze how these industries react to news data. These firms are:

1. **MSFT**: Microsoft Corporation, as the representative of tech firms (4.13% share).
2. **JNJ**: Johnson & Johnson, as the representative of goods manufacturing companies (1.53% share).
3. **JPM**: JPMorgan Chase & Co., as the representative of the financial sector (1.52% share).

3.1.2 Getting news data from reuters

Contrary to financial information which follows a tabular structure, it is relatively easy to handle and there are not many barriers to its obtention; news data is presumed to be a more precious resource on account of the fact that the information is highly dispersed, extremely unstructured and requires certain expertise for manipulation. In addition, the potential of these types of data like news, market trends or online opinions is believed to be gigantic since it contains lots of relevant information which, if properly treated, can be critical in a large number situations. Still nowadays the problem with news data is its complexity as it is unformatted text posted somewhere in the net with little preparation to be retrieved and used for data analysis. In financial analysis, news play the role of alternative data aiming to capture additional insights out of reach for traditional financial data.

As expected, several developers have been working on APIs that deal with previously mentioned complexity and provide the user with cleaned and preprocessed data almost ready to be fitted in a math model. Enterprise and commercial projects

⁵ If feasible a good option is to include periods belonging to different economic cycles, for instance information within and out the context of 2008's economic crisis.

that require news data in their analysis use to benefit from these APIs like the Google News API to search and get live headlines, articles, images, and other news meta-data. The negative side is that to make their work profitable, these API force the use to pay for their services. The pricing is variable but, in general, without paying, data will very likely flow in with delay and the amount of requests per day will be limited to a small amount. For the purposes of this analysis, using the free services of APIs is not feasible and a lower-level programmatic approach is required. Because of the inaccessibility of APIs, the complexity of the retrieval has increased exponentially and again, parsing issues and html keyword findings will have to come to the stage.

Before going into detail with html parsing, different utilities are defined to ease the process and perform tasks that are to be required repeatedly.

```

1 import numpy as np
2 import time
3 import datetime
4 from bs4 import BeautifulSoup
5 from urllib.request import urlopen
6
7 def get_soup_with_repeat(url, repeat_times=3, verbose=True):
8     for i in range(repeat_times): # repeat in case of http failure
9         try:
10            time.sleep(np.random.poisson(3))
11            response = urlopen(url)
12            data = response.read().decode('utf-8')
13            return BeautifulSoup(data, "lxml")
14        except Exception as e:
15            if i == 0:
16                print(e)
17            if verbose:
18                print('retry ... ')
19            continue
20
21 def generate_past_n_days(numdays):
22     """Generate N days until now"""
23     base = datetime.datetime.today()
24     date_range = [base - datetime.timedelta(days=x) for x in range(0,
25     numdays)]
26     return [x.strftime("%Y%m%d") for x in date_range]

```

LISTING (3.5) News data scraping utilities

The first function outputs a soup object, similar to XML format, from a url link input and repeats the operation a predefined number of times (3 is the default) in case of http failure due to connection problems or simply because the webpage did not have enough time to fully load. One interesting control to avoid http failures is to include `time.sleep`, a command that pauses Python execution, very useful to let the webpage time to load or to simulate human behaviour when scraping the net. The second function generates a datetime object in Python ranging N days until now.

The news case is slightly different, in terms of coding, than scraping stock data. As discussed earlier, complexity is higher and when facing complex multitasking problems, the general approach in python is to built class objects capable of performing different functions. A class object is an object constructor, an environment to create different objects that empower the class with their capabilities. To understand how classes work, it is important to pay special attention to the built-in `__init__()` function. This function is executed when the class is initialized to assign values to object properties, or other operations that are necessary to be conducted when different objects. A class object contains different methods belonging to the objects.

These methods are functions inside the class that endow the object with certain capabilities. Last important note concerns the self parameter. It is a reference to the current instance of the class, and is used to access variables that belong to the class object in a way blocks inside it will always be able to call objects defined in the init stage despite having been modified for other purposes.

```

1 import os
2 import sys
3 import datetime
4
5 import inspect
6 currentdir = os.path.dirname(os.path.abspath(inspect.getfile(inspect.
    currentframe()))))
7 parentdir = os.path.dirname(os.path.dirname(currentdir))
8 sys.path.insert(0, parentdir)
9 import utilities
10
11 def main():
12     reuter_crawler = ReutersCrawler()
13     reuter_crawler.run(Ndays)
14
15 if __name__ == "__main__":
16     main()

```

LISTING (3.6) Reuters news crawler

Listing 3.6 does a very simple job, it loads some required libraries, including the utilities module located in some parent directory and executes method run from ReutersCrawler class defined in Appendix A. Roughly speaking, the function takes a numeric input and crawls news from Reuters for a number of days in the past equal to the input given. The whole process is very complex and can be extremely costly to obtain a large dataset with news from different firms over a relatively lengthy time horizon.

ReutersCrawler starts by loading any finished routers locally available. Calc_finished_ticker method is required whenever restarting a task since it loads the already retrieved data into a new temporal object named finished.reuters, mainly for validation purposes. The same process is executed with the news_failed_tickers file and load_failed_tickers method. After generating the past dates, the ticker list file is opened and a for loop is used to read over all the lines (each line represents a different ticker).

After loading the information required from local files, essentially the ticker list if running the script for the first time, all the work is done by calling fetch_content method. Notice method run executed in listing 3.6 calls fetch_content alone from all previous python definitions in the class. The reason is because everything happens inside the fetch_content object although the method calls, at the same time, other definitions in the class that will be discussed further following the flow of execution.

Firstly, inside fetch_content, the csv with the tickers is broken into its four components: ticker, name, exchange and market capitalization. The exchange variable is included because Reuter links present the following structure:

- Domain: "https://www.reuters.com/finance/stocks/company-news" + company ticker + exchange suffix
- Example: <https://www.reuters.com/finance/stocks/company-news/MSFT.O>

Previous url looks for news on the day the script is run and, subsequently, it is passed as an argument to get_news_num_whenver, a method that checks if the

ticker has any news and, if true, returns the count. The output is stored in `news_num` and represents the number of news available on Reuters the day the script is being run. However, the goal is to retrieve past news for a given date range, and notice `fetch_content` method has a `date_range` argument hence the scraper should be capable of accessing Reuters drop down menu and look for news in the past. Besides further date concerns, up to this point, there are three different paths to follow depending on the value of `news_num`:

- If `news_num` \neq 0, `fetch_within_date_range` method is called. This object performs 5 tasks:
 - Keeps track of the number of days with missing content, initializing the value at zero and resetting it every time news are obtained (variable `missing_days`).
 - Controls for whether a retrieved html code has content or not, initializing the value at False and turning it into True every time news are obtained (variable `has_content`).
 - Loops over the date range and appends "?date=", followed by `new_time` (date index over `date_range`), to previous dateless url. This final url takes dates into consideration, the final structure of a Reuters' link for any past date is the following: <https://www.reuters.com/finance/stocks/company-news/MSFT.O?date=mmddaaaa>
 - Retrieves html associated to the previously defined url and stores it as a soup object.
 - Calls `parse_and_save_news`, a method which takes the previous soup object as input and returns the top story and the new text displayed in Reuters.

The corresponding outputs are all automatically converted to csv format and stored in files `finished.reuters`, `news_failed_tickers` and `news_reuters`.

- If the company has news (`news_num` \neq 0) on the current day, but none of them found within the given date range, the ticker is set as LOW priority and saved in a separate file to be used later, if desired. Variable `no_news_days` takes account of the consecutive number of days with no news found and it is restarted whenever a new is fetched.
- If the company has no news, even with no date range specified, the ticker is added to the LOWEST priority list.

Overall, all the methodology is oriented towards optimizing the process by first targeting webpages in which the program knows in advance that they contain useful data. Furthermore, `fetch_within_date_range` stops the crawler if a company has no news for a number of consecutive days in the past. The stop crawling limit is calculated using the number of news found in the current day, thus proportional to an indicator of news frequency. The higher the news count, the longer willingness to wait for new data to appear on Reuters. If a forced stop occurs, it is probably due to some kind of systematic error and the scraper will very rarely find more news in previous days hence the effort of trying is not worth. Finally, a company may have multiple news in a single day but as the information within a day should be highly correlated and, to simplify the process, it is perfectly valid to consider only top stories for prediction purposes.

Variable	Type	Description
Date	datetime (<i>yyyy-mm-dd</i>)	Trading date in the CRSP output calendar for the period.
High	Floating point	Highest trading price during the day.
Low	Floating point	Lowest trading price during the day.
Open	Floating point	Daily price of the first trade after the market opens.
Close	Floating point	Last available daily price before the close of the day.
Volume	Floating point	Total raw number of shares of a stock traded on the given date.
Adj Close	Floating point	Close price amended to accurately reflect the stock's value after accounting for any corporate actions (stock splits, rights offerings, dividends, etc.). It is considered to be the true price of the stock.

TABLE (3.1) Stock data: Variables

3.2 Variables

After running the code provided in previous sections, all the data needed in further steps will be already available in a local directory. Despite the files may need a bit more of processing, the data format is quite intuitive and before getting into the details of the analysis, it is always advisable to get familiar with the data. Insights such as the description of relevant variables, data types or formats are of great importance after obtaining the data and will be definitely useful when undertaking next stages in the data science process. The common approach is to conduct some EDA to get a first taste of the information we are working with. This section is the preface of subsequent EDA and it presents a brief description of the variables obtained through scraping the net.

The data employed is divided into stock market data and news data. Regarding stocks, each company is downloaded as a separate file, in this case csv, comprising the variables presented in table 3.1.

Each company included in the analysis has its associated csv file with previous list of variables. As said before, in this particular scenario the time horizon considered ranges from 2010/01/01 until the current day. The idea is to build an application capable of getting updated market data automatically by just running the script.

Regarding news data, the dataset is entirely made of text data. It has 6 manually created fields accounting for the following information:

1. **Ticker:** The ticker symbol.
2. **Company:** The name of the company.
3. **Date:** The date, still a text string but properly formatted *yyyymmdd*.
4. **Text1:** The top story for the given date (e.g., **JNJ 06/06/2019:** *"FDA advisory panel recommends approval of TB Alliance's tuberculosis treatment"*).
5. **Text2:** Full new's text in the given date (e.g., **JNJ 06/06/2019:** *"June 6 Independent experts of an FDA advisory panel voted in favor of the not-for-profit TB Alliance's"*

treatment for drug resistant tuberculosis, as a part of a three-drug combination regimen").

6. **Type:** A factor indicating whether the text is a normal new or a top story.

Notice the linking variable between the two tables is the date. Using this variable, it is possible to relate ups and downs in stock prices with the information coming from Reuters and conduct a number of interesting experiments, such as analyzing the time lag between favorable or disadvantageous news and the resulting variations in financial markets.

3.3 Exploratory Data Analysis

For many people, the EDA stage of a data science project is a bit difficult to describe in concrete terms. The goals are many, but the general idea pursued is to get a first taste of how the data looks like and prepare it for later fitting. EDA is responsible for checking if there is any clear evidence for or against an hypothesis, validating the collected data, identifying correlations between variables or certain areas in which more data is needed.

In this stage, finer details of data presentation and statistical evidence should be avoided since the focus is not necessarily the final product but the information available to work with. EDA is important for the investigator or project manager because it allows to make the first decisions such as what is interesting to follow up and which ideas are no longer feasible as the data will never provide enough evidence. This preliminar analysis is important to save resources and ensure the project moves forward, remaining within its budget.

3.3.1 Company pricing data

With respect to financial data, visualizing the time series may be highly revealing regarding the identification of certain patterns. Properties associated with time series patterns are mostly trends, seasonalities and cycles. Accordingly, it is a good start to plot adjusted close prices over the years under observation. Even though all the data is available, the information for each company is stored in a separate csv file and the objective here is to asses all the data together.

```

1 def compile_data():
2
3     with open("sp500tickers.pickle", "rb") as f:
4         tickers = pickle.load(f)
5         tickers = [ticker.replace('.', '-') for ticker in tickers]
6         tickers = [ticker.rstrip() for ticker in tickers]
7
8     main_df = pd.DataFrame()
9
10    main_df = pd.DataFrame()
11
12    for count, ticker in enumerate(tickers):
13        df = pd.read_csv('stock_dfs / {}.csv'.format(ticker))
14        df.set_index('Date', inplace=True)
15        df.rename(columns={'Adj Close': ticker}, inplace=True)
16        df.drop(['Open', 'High', 'Low', 'Close', 'Volume'], 1, inplace=
17                True)
18        if main_df.empty:

```

```

19     main_df = df
20     else:
21         main_df = main_df.join(df, how='outer')
22
23     if count % 10 == 0:
24         print(count)
25     print(main_df.head())
26     main_df.to_csv('joined_closes.csv')

```

LISTING (3.7) Joined close prices

Listing 3.7 loads the ticker list and unifies all the individual stock datasets selecting, from each company, variable 'Adj Close' and dropping the rest. Adjusted close prices are used since they reflect the true price of the stocks and are the optimal indicator of company's performance in the marketplace. Previous code loads the entire list of tickers although the resulting data frame is subsequently filtered to include only the three targeted firms. Once these prices are available, matplotlib is used to build subsequent graphs.⁶

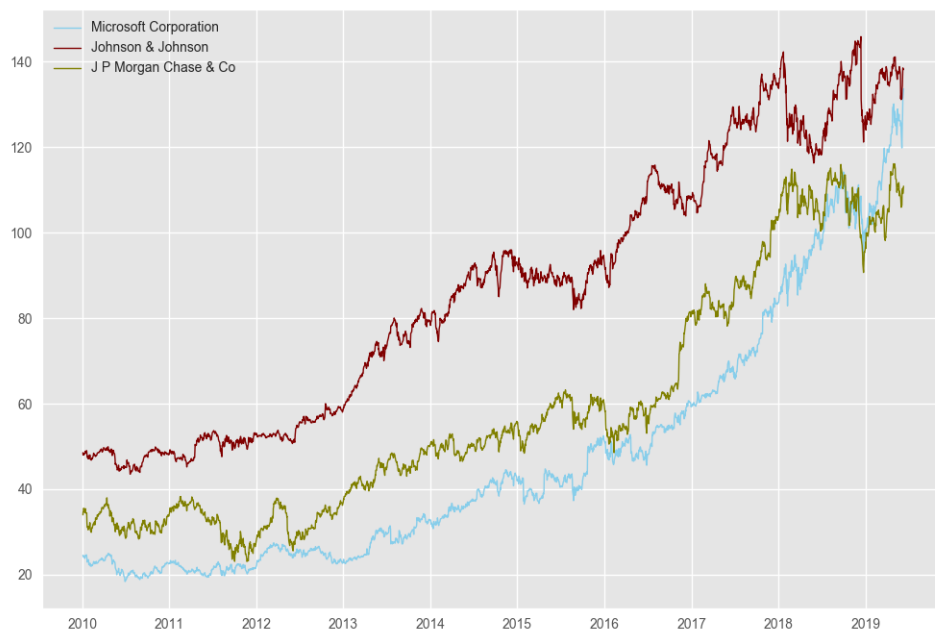


FIGURE (3.2) Evolution of Adjusted Close Prices

In terms of the previously mentioned properties of time series, neither seasonalities nor cycles are appreciated, whereas the three companies show a positive trend in their respective stock price. Their evolution appears to be aligned with the economic cycle and the growth of USA GDP. If someone looks at figure 3.2, the immediate conclusion might be that investing in any of the three firms is a good decision since long term profits seem reasonable and stocks are not specially volatile. However, the outcome of a hypothetical investment is not that straightforward to assess as many considerations such as the time value of money, fees, taxes, opportunity cost of capital, etc. should not be forgotten. Despite financial details are out of the scope of this report since the focus is predominantly on prediction matters, previous argument seeks to support the idea that trading strategies shall chase opportunities

⁶Matplotlib is a popular plotting library in Python which handles different data structures and produces high quality figures in a wide variety of formats, sometimes allowing interaction. See [Matplotlib](#) for further details.

in short time intervals and anticipate ups and downs to get a real benefit from stock price variations.

Another important magnitude is the volume of stocks traded in a given day since prices in stock markets fluctuate a lot depending on the supply and demand of securities.

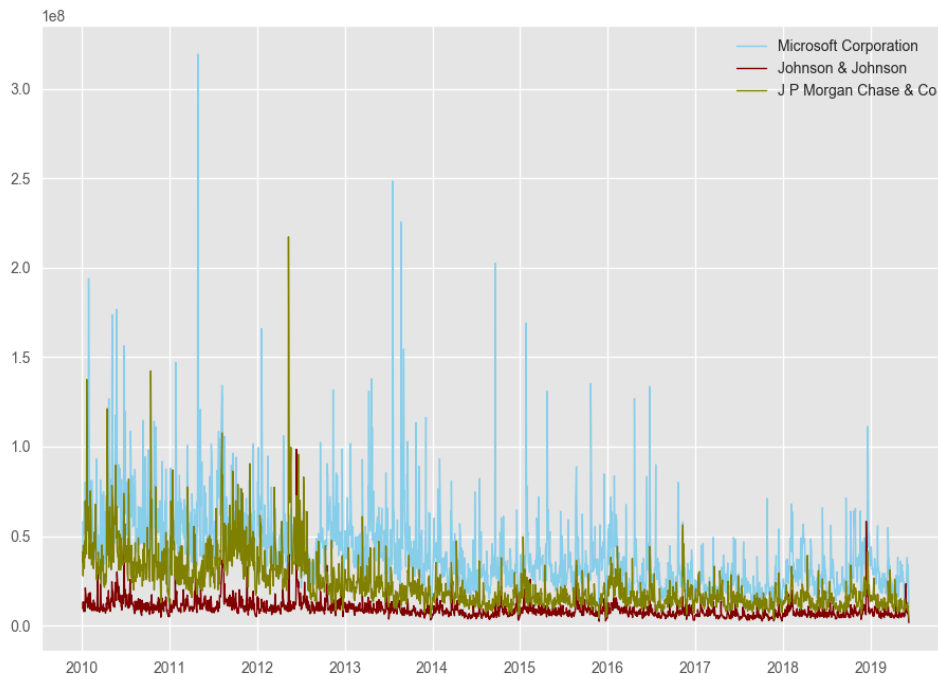


FIGURE (3.3) Volume traded (in hundreds of millions)

Figure 3.3 illustrates the volume of stocks traded throughout the time horizon for the three companies. The time series of volume is generally characterized by several peaks associated with massive buying or selling actions. These peaks usually appear after favourable or disadvantageous news releases, depending on whether the order is a buy or a sell. For instance, the Microsoft Corporation experienced a noticeable peak on 29 April 2011. Notwithstanding the news database will be discussed further below, a couple of top stories from the day before Microsoft's volume peak are displayed to analyze its content and see if they can shed light to this sudden increase.

- 2011/04/28: U.S. top court declines to hear Microsoft antitrust case.
- 2011/04/28: U.S. UK advise avoiding Internet Explorer until bug fixed

Apparently, the two press releases are bad news and, although returns on Microsoft stocks will unveil the real effect on prices, signs until now suggest a drop. It is natural to think that news releases and announcements have a major influence on abrupt price variations. Negative or controversial information about a certain company increases uncertainty, generating higher variability in stock prices, since the future scenario is unclear, and therefore makes the investment more risky.

Investors generally speculate with variability on the basis of different types of analysis but, in all cases, the unit of study are the returns, which they try to maximize. Stock market returns are returns generated out of the stock market, in the form of profit through trading or dividends perceived from the company. In reality,

investors do not pay attention to the nature of the return obtained since adjusted close price reflects both, and the bottomline is merely to make the highest profit possible no matter whether it is achieved through dividends or by trading stocks. In this respect, further EDA will focus exclusively on returns for each company under study since they are the value reference for securities investment and the magnitude to be maximized.

Pandas has a pre-build method called `pc_change()` which, when applied to a pandas dataframe field, creates a for loop that goes and calculates for each row the percentage change with respect to the previous observation. By applying the aforementioned method to 'Adj Close' it is quite simple to obtain a column representing return values.

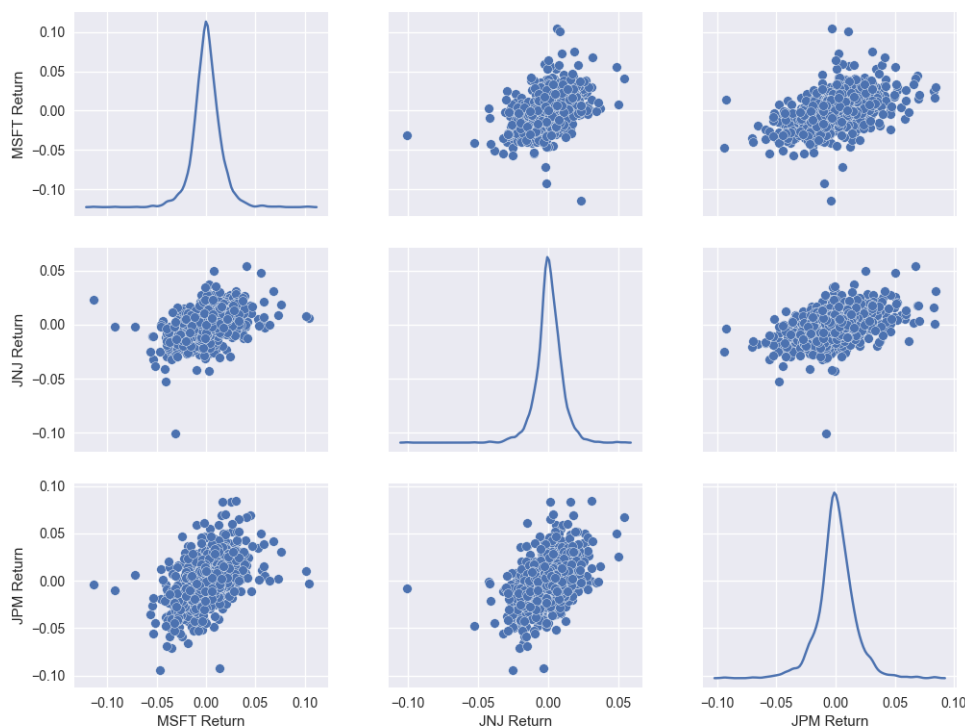


FIGURE (3.4) Distribution of stock returns

Figure 3.4 displays joined relationships (biplots) and univariate distributions of returns for Microsoft Corporation, Johnson & Johnson and J P Morgan Chase & Co throughout the time period considered. In the three cases, the distribution of returns resembles a normal distribution and correlations are positive although more evident between Microsoft Corporation and J P Morgan Chase & Co. Results are not surprising since, although belonging to different sectors, the three companies are large American firms and they have lots of things in common. In general, if one of them is doing good in the market, the others will very likely be doing so.

Tables 3.2 and 3.3 gather, respectively, the dates when companies registered their highest and lowest returns. Once again, it may be a good exercise to explore news headlines around the dates shown in previous tables to detect if any positive or negative statements originated these 'abnormal' returns. Furthermore, previous tables also reveal that negative returns are, in absolute terms, higher than the positive ones and the range of returns for the Microsoft Corporation is much more wider, in both sides, than the rest.

Symbol	Company	Date	Return
MSFT	Microsoft Corporation	2015-04-24	10,45%
JNJ	Johnson & Johnson	2011-08-11	5,38%
JPM	J P Morgan Chase & Co	2011-11-30	8,44%

TABLE (3.2) Highest stock returns

Symbol	Company	Date	Return
MSFT	Microsoft Corporation	2013-07-19	-11,40%
JNJ	Johnson & Johnson	2018-12-14	-10,03%
JPM	J P Morgan Chase & Co	2011-08-08	-9,41%

TABLE (3.3) Lowest stock returns

Symbol	Date	Return	News
MSFT	2013-07-19	-11,40%	Microsoft profit misses as Surface tablets languish; shares drop.
MSFT	2015-04-24	10,45%	Microsoft profit revenue beats Wall Street view; shares up.
JNJ	2018-12-14	-10,03%	J&J shares drop on report company knew of asbestos in baby powder.
JNJ	2011-08-11	5,38%	FDA approves Gilead's once-daily HIV pill.
JPM	2011-08-08	-9,41%	No news available
JPM	2011-08-08	8,44%	No news available

TABLE (3.4) News from the day before abnormal returns

Symbol	Company	Standard deviation
MSFT	Microsoft Corporation	0,014488
JNJ	Johnson & Johnson	0,009343
JPM	J P Morgan Chase & Co	0,016000

TABLE (3.5) Stock returns standard deviation

Symbol	Company	Standard deviation
MSFT	Microsoft Corporation	0,017805
JNJ	Johnson & Johnson	0,013796
JPM	J P Morgan Chase & Co	0,013931

TABLE (3.6) Stock returns standard deviation in 2018

Ponder to what extent an immediate drop or increase in share prices might be anticipated if investors make a proper use of news data. Table 3.4 discloses that press releases are great indicators of what will happen with share prices. Up to this point, it does not seem unreasonable to assume that by detecting media trends and developing methods capable of assessing the current scenario surrounding a company, variation in stock prices is very likely to be predictable.

Nevertheless, variation always involves certain risks, in this case the risk of suffering losses. In EDA it is also recommended to make an initial directional assessment of variation in the data. There are several statistical figures for measuring variability but, among all, standard deviation has probably the most direct interpretation. As stated in Table 3.5, JPM shares present the highest standard deviation, hence assumed to be the riskiest investment.

More visualizations can focus on specific periods with presumably relevant events. For instance, figure 3.2 exhibits high volatility of adjusted close prices during 2018 thus, it may be interesting to scale up previous graphic and analyze the distribution of company's returns and their respective standard deviations within the mentioned timespan.

Apparently, the Microsoft Corporation is the company that offers potentially higher returns during the period, although its variability is presumably the largest as well. J P Morgan Chase & Co bears the brunt in that particular year since its returns' distribution is centered slightly before 0 and, as shown in Table 3.6, its stocks do not provide any significant advantage over other companies in terms of volatility.

Analysis conducted thus far do not evince major changes in stock prices over the time horizon considered. Adjusted close prices show clearly positive trends, and returns are considerably stable (standard deviations around 1pp) and not specially distant from 0. The previously described picture is common when working with big multinational firms on account of a strong brand reputation, operational diversity and easy access to financing. In consequence, their stocks are highly stable overtime and the absence of risk in finance is not rewarded with high returns. A priori, investing in any of the companies under study may be considered a safe investment.

Nevertheless, this is not always the case and investors quite often encounter stocks whose mid/long term trend is not as evident as in the case of Microsoft, Johnson & Johnson or J P Morgan. SMA is a widely used trend indicator in finance, calculated by taking the average of stock prices over certain previous periods on a

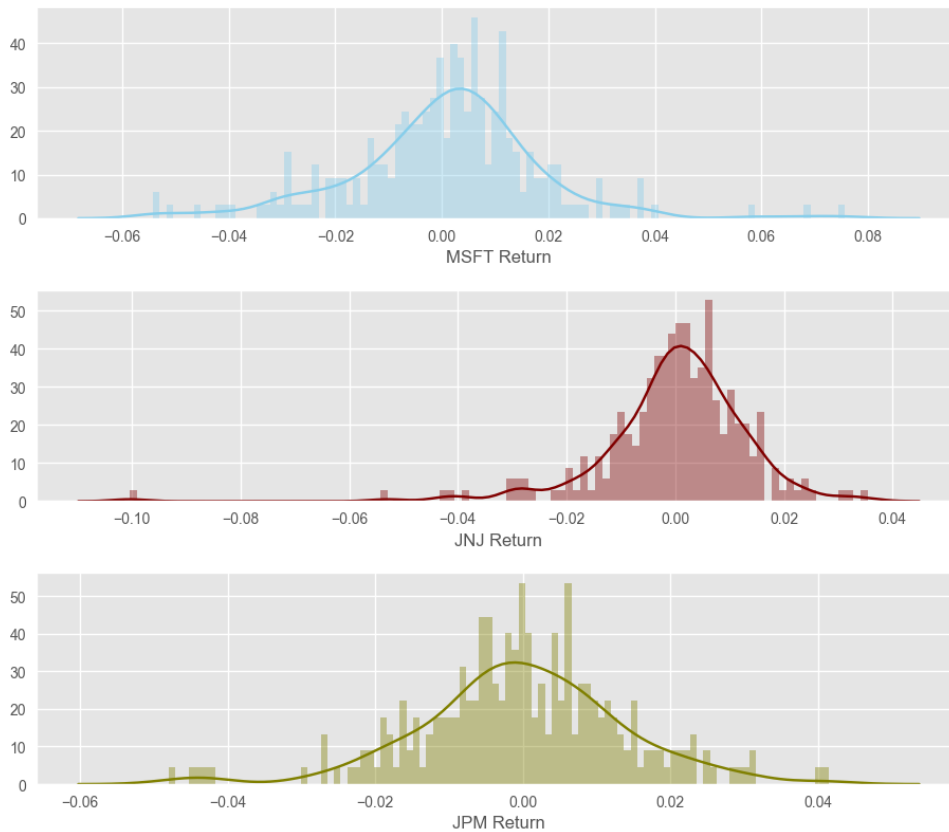


FIGURE (3.5) Distribution of stock returns in 2018

rolling window. The SMA framework employs two lines, a short-term moving average and a long-term moving average, and the idea is to make trading decisions when both lines cross.

Theory suggests that if 50-day moving average crosses above the 200-day moving average, the crossover is bullish and interpreted as a sign of buying. On the contrary, if the 50-day moving average crosses below the 200-day moving average, the crossover is bearish and the investor should sell the security. SMA is very popular in technical analysis since financial market technicians assume stock prices move in trends. This trading discipline focuses on the actions of the market itself rather than the goods in which the market deals. Besides SMA, many other indicators are used in technical analysis to evaluate investments and, although the discipline's scope is different than pure predictive modelling, some of the elements are still suitable for EDA ⁷.

Another interesting visualization regarding stocks data involves building a correlation table to unveil any interesting correlated data and set the stage for further clustering. Although not of great relevance here, since only three companies are considered for modelling purposes, correlation matrices are extensively used in finance and multivariate analysis when dealing with a large number of variables and they definitely deserve a mention. Correlation tables are nothing but handy ways of reporting correlation coefficients between variables. In this regard, one can calculate correlation coefficients for both adjusted close prices and returns.

⁷ See (Kirkpatrick II and Dahlquist, 2010) for more information on technical analysis

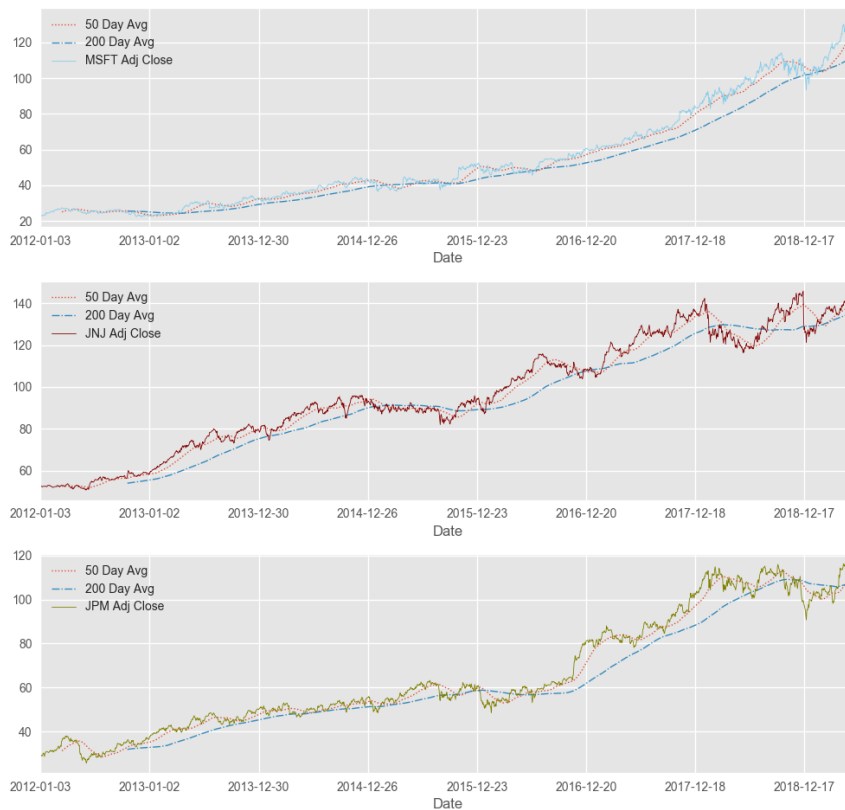


FIGURE (3.6) SMA, 2012/01/01 - 2019/06/10

Either a heatmap or a cluster map is a good option for representing correlation matrices. Figures 3.7 and 3.8 are aligned with previous insights showing that the relation between Microsoft and JP Morgan stocks is stronger than with Johnson & Johnson. With respect to coefficients, they are all positive (in both cases) although notably higher in figure 3.8. A naive explanation behind this phenomenon might be the fact that stock prices are influenced by the market to a greater extent than returns, where company's performance might play a more important role.

Overall, it is expected that firms from the same index, or those who share common characteristics, are positively correlated. Ideally, investing in a bunch of companies with zero correlation over time would be a decent way to be diverse, although it is still early to assume companies with coefficient values close to zero are not related at all.

Lastly, another significant chart type in financial analysis is the candle plot or candlestick chart. It is used to characterize price movements of some financial magnitude. Each "stick" represents the four relevant price fields of a single day: high, open, close and low. The filled part is popularly known as the real body and it may appear in two colors; one for those days in which the close price was higher than the open price, and the other for the opposite situation. In general, candlestick charts are an excellent representation of trading patterns over short time periods and they are widely used to foresee the short-term direction of the price.

Compared to traditional line or bar charts, candlestick charts are easier to interpret and more visually appealing. Each stick represents a day and the comparison between open and close prices as well as high and low is instantaneous. Furthermore, the mapping of relevant dates to news data is also direct. As an example,

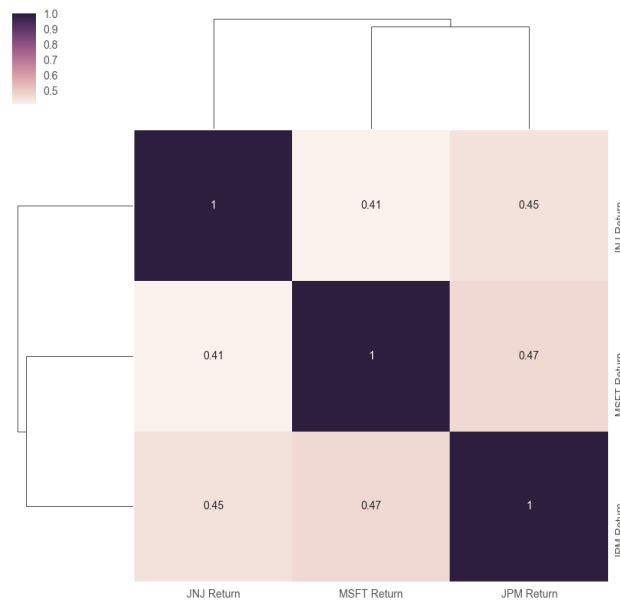


FIGURE (3.7) Stock returns cluster map

figure 3.10 shows a relevant drop and large price variations, in JNJ stocks during the first half of May. News found around these days are the following and, once again, they justify the negative impact on stock prices.

- **13/05/2019:** An Oklahoma judge on Monday cleared the way for Johnson & Johnson and Teva Pharmaceutical Industries Ltd to face trial at the end of May in a lawsuit by the state's attorney general accusing the drugmakers of helping fuel the opioid epidemic.
- **07/05/2019:** Johnson & Johnson agreed to pay about \$1 billion to resolve the bulk of lawsuits claiming the company sold defective metal-on-metal hip implants that ultimately had to be removed Bloomberg reported on Tuesday citing people with knowledge of the matter.

Summarizing, the most relevant conclusions obtained from the EDA regarding stocks are that the three companies present relatively stable prices with positive trends and the distribution of returns is largely normal centered around zero with not very large variances.

3.3.2 News data

The EDA approach to the news data is totally different since the information is mainly text. When confronted with a promising yet large text file which contains relevant features and data subsets that might lead to prominent findings, but also tones of useless information, it is necessary to, somehow, categorize and quantify the text under consideration. NLP covers a broad range of techniques that apply computational analytical methods to textual content, which provide means of categorizing and quantifying text that can help researchers explore their textual data

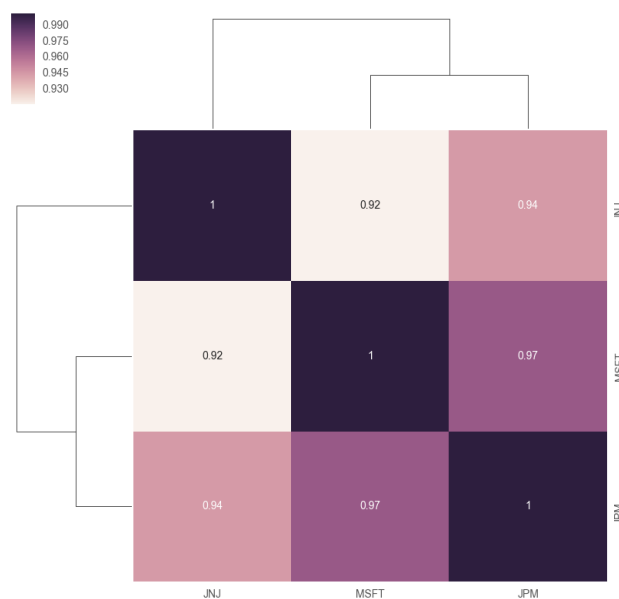


FIGURE (3.8) Adj close price cluster map

(Saldaña, 2018). The vast majority of these techniques are of exploratory nature since they pursue to 'explore' texts aiming to find indications that might be worth investigating further. Nevertheless, since NLP techniques have a strong mathematic component and are formally defined, they are regarded as belonging to the methodology section rather than to EDA according to the definition provided in the beginning of the chapter.

Having said the above, after running the scraper it is interesting to see how many news have been retrieved from each stock and the associated date ranges.

Symbol	Company	News count	Date range
MSFT	Microsoft Corporation	3.415	2011/07/06 - 2019/06/17
JNJ	Johnson & Johnson	1.681	2011/07/07 - 2019/06/06
JPM	J P Morgan Chase & Co	1.780	2015/02/17 - 2019/06/13

TABLE (3.7) Number of news retrieved from Reuters

Microsoft doubles the other two companies in number of associated news within the date range. However, the date range for J P Morgan is substantially narrower thus, the density of news is presumably higher for the bank. Recall the news scraper is programmed to stop the crawling if no news are found in some consecutive number of days in the past. In this case, after 2015/02/17, the scraper did not find any news on Reuters associated with J P Morgan for 71 consecutive days, hence the crawling stopped and JPM was classified as lowest priority. Further requests to J P Morgan news can be ordered for days before 2015/02/17 if necessary but, the problem appears to be systematic (change in ticker nomenclature, copyright problems...) since it is quite rare that a company with such a high news density has no

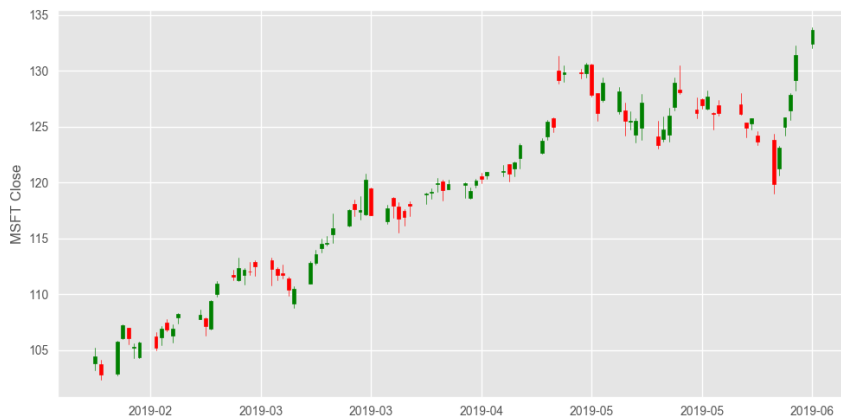


FIGURE (3.9) Candlestick chart MSFT - 2019

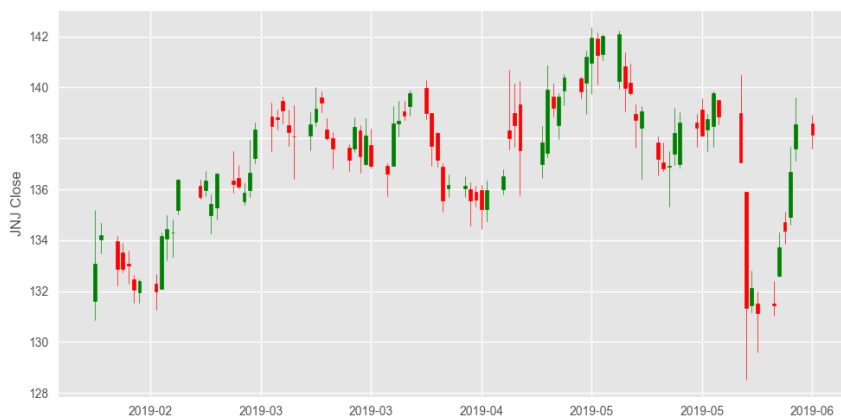


FIGURE (3.10) Candlestick chart JNJ - 2019

associated news for 71 consecutive days.

The previously mentioned lack of data is not a problem here as news act as an enriching factor to company pricing data added to the fact that sentiment analysis and feature engineering offer methods to deal with different sample sizes. Next step, concerns wordcounts of headlines and regular news. There are not solid arguments to think that wordcounts may vary between different companies but they are an interesting figure that will be useful in further stages.

Symbol	Company	Headlines	Regular news
MSFT	Microsoft Corporation	9,779298	32,348007
JNJ	Johnson & Johnson	9,966854	34,286517
JPM	J P Morgan Chase & Co	8,971596	31,151098

TABLE (3.8) Average wordcount of headlines and news

As presumed, average wordcount values are very similar for the three companies. Before getting into the details of sentiment analysis and feature engineering, it is interesting to see which are the most popular words in our news database. Figures 3.12 and 3.13 exhibit the top words in both headlines and news by means of a



FIGURE (3.11) Candlestick chart JPM - 2019

wordcloud, probably the most common figure to represent word frequencies in text data.

At a glance, it is observed that the words in both clouds are substantially different. Top stories ought to capture the most popular news of the day or, somehow, provide a general outline of everything involving the company, whereas news can be thought of as individual stories reflecting the reality. Very rarely the same words are used for both purposes and it is expected to detect discrepancies in most frequent words between the two fields. Nevertheless, the information contained in both news and headlines is expected to be similar with respect to the measures used in text analytics. For instance, when quantifying the postiveness/negativeness of news for a particular company and day; the information in headlines and news will probably be aligned as it is extremely unusual for a news website to write a positive headline when news are negative, and viceversa. Accordingly, futher analysis are perfectly applicable to either case.

In addition, the process of converting text data to something readable for a computer might require some pre-processing as well. In NLP, it is highly common to remove useless information by means of a 'stop words' list which the computer is programmed to ignore. These words are generally commonly used words (such as 'the', 'a', 'an', 'in') that are theoretically equally distributed among the units of study. Nltk in Python offers a pre-built easy-to-download list of stop words although is preferable to obtain a industry specific set if available. In this paper, the list of stop words presented in Appendix B has been obtained using some discrepancy measure from a NN trained with S&P500 data. Its use remains only for exploratory purposes since tokenization and feature engineering methods are not employed in the modelling stage. The interest relies exclusively on polarity scores and the methodology employed will automatically deploy these 'stop words' without requiring any specifications.

As discussed earlier, sentiment analysis is not treated in this section since the method is formally defined and uses data transformations presumed to be beyond the scope of EDA.

Chapter 4

Methods

The question raised in this paper is mainly of predictive nature. One of the major objectives pursued, besides automating the data obtention process, involves being able to cast future stock prices of companies using time series of financial data and information from news on Reuters. Notwithstanding the theoretical nature of the study, the capacity of anticipating future ups and downs in securities' price is a precious resource that many companies use to get profits in stock markets or make business with it.

Traditionally, financial analysis has been highly based on a large list of financial indicators and data taken from financial statements. Accounting ratios, percentage of inter-firm comparison, intra-firm comparison, common size statement, etc., were very popular figures employed to ascertain relationships between internal accounting magnitudes and what was happening in the market. Even though these methods allow some space for certain mathematical modelling, the approach was still highly theoretical-based and strongly rooted in economic theory.

In recent years though, Machine learning and Deep learning have proven to be effective strategies to maximize profits in financial markets. The approach is completely different since financial statements are no longer contemplated and the focus now is on the alternative sources of information which provide additional insights to capture drivers outside the scope of economic theory but still influential in stock markets. Finance is highly nonlinear and some theorists even believe price movements are completely random. In 1973 professor Burton Gordon Malkiel published his famous '*Random Walk Down Wall Street*', one of the most famous experiments about the reliability of investments made by stock market analysts. In his book, Malkiel compared an investment guided by professionals with another made completely at random, precisely, he used as a figure of speech, a blinded monkey throwing darts to a wall street newspaper with stock names. Surprisingly, the monkey's portfolio got higher returns than 85% of professional investors in addition to exceeding returns achieved by the main indexes. By that time, traditional time series methods like ARIMA or GARCH models were the dominant note in the industry, and these methods are effective only when the series is stationary. Stationarity of time series is a restricting assumption that requires data to be preprocessed by taking log returns (or other transformations). Furthermore, even if the data shows certain stationarity, in a live trading system there is no guarantee of stationarity as new data is added. Neural Networks partially resolve the stationarity problem as they do not require any data assumptions and, as discussed in chapter 1, they are also very good at finding patterns out of raw data.

Flexibility and non-linearity are probably the most powerful characteristics of neural networks. They do not require any data assumption and are capable of approximating practically anything that can be turned into a number. Another important feature of neural networks is their adaptive feature/basis structure which

enables them to face problems with many inputs and high dimensionality, such as images. Furthermore, even though the training can be tedious, once finished, predictions are instantaneous. On the other hand, likelihood functions are no longer a convex function of the model parameters hence the problem might become to expensive and time consuming to train with traditional CPUs. In addition, overfitting and generalization problems are very likely to appear, specially in data scarce scenarios.

The methodology employed in this paper is strongly based on Deep Learning and Neural Networks seeking to turn a financial/economic problem into pure predictive modelling and forget about any data assumption which is out of our control. The hypothesis raised pretends to test if time series modelling of financial data can be enriched using news' sentiments. Some neural networks achieved significant results in time series prediction of financial data, although the methods employed are extension of regression algorithms that can only handle quantitative variables. Accordingly, the inclusion of any kind of sentiment magnitude in these models has been a real challenge still unresolved. The common approach to stock price prediction using news data is slightly different as it generally uses classification algorithms on labelled data and the predictions are predominantly buy or sell signals. On the occasion of the recent emergence of VADER sentiment analysis, a method capable of assigning quantifiable sentiment scores to text data, this study aims to test whether the VADER methodology has practical applications on the modelling of financial time series from a regression perspective, essentially acting as an enriching factor to other quantitative magnitudes considered.

Firstly, to turn news data into something a computer can understand and use when training the network, VADER sentiment analysis is conducted to assign a sentiment score to each new in the dataset. Subsequently, company price data is merged with the previously obtained scores by date, and different models are fitted. There is still little theory behind deep learning models hence highly difficult to find answers to questions like: which NN architecture is the best for this problem, what is the optimal number of epochs/layers or which activation functions should be considered. Sometimes NN design is more an art than a science, with trial and error as the best option to assess a model's performance. Accordingly, the models built seek to provide the reader with certain educated guesses that might be of practical utility in the industry based on previous empirical evidence.

4.1 Theoretical considerations

Neural Networks are a relatively new instrument in scientific computing and they offer many advantages over traditional inferential methods, some of them already discussed. However, it is also important to be concerned about their limitations and the theoretical basis behind. Some of the most important ones are presented below.

4.1.1 Universal approximation theorem

The universal approximation theorem lays the mathematical foundations of artificial neural networks by stating that a feed-forward network (no cycles) with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n .

One of the first versions of the theorem was developed by (Cybenko, 1989) for sigmoid activation functions.

Theorem 1. *Let σ be any continuous discriminatory function. Then finite sums of the form:*

$$G(x) = \sum_{j=1}^N \alpha_j \sigma(y_j^T x + \theta_j) \quad (4.1)$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum, $G(x)$ of the above for which

$$|G(x) - f(x)| < \epsilon, \quad \text{for all } x \in I_n$$

Previous expression clearly resembles the mathematical representation of an artificial neuron in 1.1. Later proofs demonstrate the equality still holds when replacing ϵ with any general nonconstant, bounded and continuous function: $\varphi : \mathbb{R} \rightarrow \mathbb{R}$.

In 2017, Hanin improved previous result 4.1 showing that ReLU neural networks with width $n + 1$ are sufficient to approximate any continuous convex function of n -dimensional input variables (Hanin, 2017).

Theorem 2. *For any Lebesgue-integrable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ satisfying $\{x : f(x) \neq 0\}$ is a positive measure set in Lebesgue measure, and any function \mathbb{F}_A represented by a fully-connected ReLU network A with width $d_m \leq n$, the following equality holds:*

$$\int_{\mathbb{R}^n} |f(x) - \mathbb{F}_A(x)| dx = +\infty \int_{\mathbb{R}^n} |f(x)| dx \quad (4.2)$$

The evolution of the universal approximation theorem through the last 20 years is a strong argument for believing that neural networks are capable of approximating practically anything that can be turned into a quantifiable magnitude without any data assumptions.

4.1.2 Activation functions

In the flow of a network, as its name suggests, the activation function is responsible for deciding whether the neuron is 'firing' or not. Consider the neuron in 1.1. It basically calculates a weighted sum of its input, adds a bias, and function θ decides whether it should be active or not.

The value of $x_i[t]$ can be anything in the range $(-\infty, +\infty)$ as the neuron itself does not establish any bounds of the output. Activation functions are added to assess the value produced by a neuron and decide whether outside connections should consider this neuron as 'activated' or not.

First thing that might come to someone's mind is a threshold based step function to discriminate between active and inactive neurons. Something such as $f(x_i[t]) = 1$, hence 'activated' if $x_i[t] > \text{threshold}$, and $f(x_i[t]) = 0$ otherwise, is very intuitive and even of practical utility for a binary problem. However, when facing a multi-class classification problem or a continuous target, step functions have various limitations as a 0/1 output does not say much about results the network is trying to predict. In this situations, it is better to obtain a range of activations and decide which neurons are firing by taking the max (or softmax) among, for instance, the values of those sets of connected neurons where at least one is supposedly firing. This lead us to conclude that, in theory θ can be any function since the operation is just a mapping of the resulting values into some domain where max or softmax functions can be later applied.

Selecting the optimal activation function is, to a great extent, an unresolved problem. If certain characteristics of the function approximated are known in advance, it is easier to choose an activation function which approximates it faster leading to a faster training process. As an example, sigmoid works well for classifiers as the function is nonlinear, smooth and 'step function like', thus tends to bring the activations to either side of the curve and makes clear distinctions on different classes. Other activations, for instance ReLU, are easier to operate with and lead to a faster training process and converge. In fact, ReLU activation functions are the most used right now since convergence is relatively fast and (Hanin, 2017) demonstrated that they are sufficient to approximate any continuous convex function of n-dimensional input variables.

Linear, sigmoid and tanh activations are covered in more detail since they are the only ones applied in the models developed in further sections.

- **Linear activation function:** It is a simple straight line where the activation is proportional to the input. The main concern with using this type of activation function is that the derivative with respect to x is constant hence the gradient has no relationship with the input data (constant gradient). This type of activation is generally applied in the last layer where no further optimization is required.
 - Equation: $f(x) = cx$.
 - Derivative (with respect to x): $f'(x) = c$.
 - Range: $(-\infty, +\infty)$.
 - Order of continuity: C^∞ .
 - Monotonic: Yes.
 - Derivative monotonic: Yes.
 - Approximates identity near the origin: Yes.
- **Sigmoid activation function:** It is a "step function like", non-linear in nature and with a smooth gradient. Narrow X intervals near the origin are very steep, which means the function has the tendency to bring Y values to either sides of the curve showing great discrimination properties and optimal performance in classification problems. However, the function is very robust to changes in X values far from the origin, a property that usually generates a problem known as 'vanishing gradients', which will be discussed in further sections.
 - Equation: $f(x) = \frac{1}{1+e^{-x}}$.
 - Derivative (with respect to x): $f'(x) = f(x)(1 - f(x))$.
 - Range: $(0, 1)$.
 - Order of continuity: C^∞ .
 - Monotonic: Yes.
 - Derivative monotonic: No.
 - Approximates identity near the origin: No.
- **Tanh activation function:** It is a scaled sigmoid function $\tanh(x) = 2 \cdot \text{sigmoid}(2x) - 1$ and it has similar characteristics. One point worth to mention is that its gradient is stronger than the sigmoid case (steeper derivatives).
 - Equation: $f(x) = \frac{2}{1+e^{-2x}} - 1$.

- Derivative (with respect to x): $f'(x) = 1 - f(x)^2$.
- Range: $(-1, 1)$.
- Order of continuity: C^∞ .
- Monotonic: Yes.
- Derivative monotonic: No.
- Approximates identity near the origin: Yes.

4.1.3 Optimization

Learning from data involves finding the unknown rules that will convert inputs into outputs. In other words, the model learns a function f , such that $f(X)$ maps to y by adjusting the model parameters. In neural networks, these parameters are combinations of weights and biases which are repeatedly updated to approximate the target variable for all training inputs. After the training stage, once the model has learnt the optimal values for the parameters, they can be used to predict values of y given new X s. Next immediate question is how does the program check the goodness of some given parameters and also, how are they updated in the proper direction to finally reach out the optimum.

Recall the focus in Deep Learning is on learning from examples. Accordingly, if the goal is to approximate y as accurately as possible using the examples available, it is necessary to define a loss function that somehow quantifies how wrong different values of weights and biases approximate the dependent variable. Even though the loss function varies from model to model, the general idea is to express it as a distance/difference between the predicted values and the actual values.

Quadratic loss functions such as MSE or MBE are often used in regression problems to measure the difference between estimated values of y and the 'ground truth' (real values of y in the training dataset) ¹.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (4.3)$$

Previous expression of the MSE 4.3 is generally adjusted to simplify calculations down the track. As long as the loss function remains as a measure of discrepancy between predictions and the 'ground truth' any modifications can be made. Let $\Theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, \dots, W^{(L)}, b^{(L)}\}$ be the set of learnable parameters, and L the number of layers in the network:

$$L(y|\Theta, x) = \frac{1}{2m} \sum_{i=1}^m (h_{\Theta}(x^{(i)}) - y^{(i)})^2 \quad (4.4)$$

Here, Θ encloses the collection of all weights and biases in the network (also called learnable parameters), m is the number of training inputs and h_{Θ} is the output from previous connected layers in the network. Notice $L(y|\Theta, x)$ is constantly modified with parameter updates but the function is entirely non-negative, since every term in the sum is non-negative. In addition, the output value of the loss

¹ Classification losses, however, measure the number of correct labels over the total number of inputs by some safety margin hence algebraic distances are no longer applicable and other functions such as cross entropy loss or hinge loss are needed.

function becomes small when network predictions resemble y values in the training dataset, suggesting that the learning has been successful.

The learning process consists in finding combinations of weights and biases that minimize the loss function through the different layers of the network. As in any situation which demands minimizing or maximizing a function, optimization algorithms are used to achieve this purpose. Loss functions are like objective functions in optimization problems, dependent on the model's learnable parameters (weights and biases), which are used to compute the target values from predictors.

Gradient descent is a general optimization algorithm widely used in NN training. Gradients enable the model to determine the direction it should take in order to reduce the loss function value. What gradient descent does is mainly evaluating how the loss function changes when introducing small variations in the learnable parameters. Mathematically, the gradient of the loss function L_{Θ} is defined as the vector of partial derivatives with respect to Θ .

Hence, if $\Theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, \dots, W^{(L)}, b^{(L)}\}$ is the set of learnable parameters, and being L the number of layers in the network, after each iteration the loss function changes as follows:

$$\Delta_{\Theta}L_{\Theta} \approx \frac{\partial L_{\Theta}}{\partial W^{(1)}} \cdot \Delta W^{(1)} + \dots + \frac{\partial L_{\Theta}}{\partial W^{(L)}} \cdot \Delta W^{(L)} \quad (4.5)$$

In addition, equations 4.6 and 4.7 display the gradient vector and the vector of changes in weights.

$$\nabla_{\Theta}L_{\Theta} \equiv \left(\frac{\partial L_{\Theta}}{\partial W^{(1)}}, \dots, \frac{\partial L_{\Theta}}{\partial W^{(L)}} \right)^T \quad (4.6)$$

$$\Delta W \equiv (\Delta W^{(1)}, \dots, W^{(L)})^T \quad (4.7)$$

Modifying the weights in the direction of the gradient will calculate $\Delta_{\Theta}L_{\Theta}$ iteratively through each hidden layer. However, changes in the loss function should be negative, reducing the value of the loss in each iteration as much as possible. Therefore, the optimizer has to figure out how choose ΔW so as to make $\Delta_{\Theta}L_{\Theta}$ negative. With previous definitions, the gradient descent in 4.6 can be rewritten as:

$$\Delta_{\Theta}L_{\Theta} \approx \nabla_{\Theta}L_{\Theta} \cdot \Delta W \quad (4.8)$$

being the gradient the mathematical object relating changes in weights to changes in the loss function. The real benefit of 4.8 is that it allows ΔW that always result in negative $\Delta_{\Theta}L_{\Theta}$. In particular:

$$\Delta W = -\eta \nabla_{\Theta}L_{\Theta}, \quad (4.9)$$

where η is a small predefined positive parameter, known as the learning rate that measures how quickly the optimizer moves towards the minimum. Replacing equation 4.9 in equation 4.8 results in $\Delta_{\Theta}L_{\Theta} \approx -\eta \|\nabla_{\Theta}L_{\Theta}\|^2$ and, as $\|\nabla_{\Theta}L_{\Theta}\|^2 \geq 0$,

it can be guaranteed that $\Delta_{\Theta}L_{\Theta} \leq 0$. Consequently, parameters will be updated as follows:

$$\begin{aligned}\Theta' &:= \Theta - \eta \nabla_{\Theta} L_{\Theta} \\ \Theta'_j &:= \Theta_j - \eta \frac{\partial}{\partial \Theta_j} \nabla_{\Theta} L_{\Theta}\end{aligned}\tag{4.10}$$

As the model iterates, it gradually converges towards the minimum of the loss function where further modifications of the parameters produce little changes in the loss. In addition, better estimates of weights and biases are also obtained in each iteration. Speed of convergence usually depends on the function the network is approximating and the learning rate η , which is tuneable and specified before the training. Excessively large learning rates might overshoot a minimum while values too small will delay convergence. An optimal trade-off should be found although a learning rate around 0,2 is considered standard and can be applied, at least as an initial guess. In recent years though, almost all neural networks use the Adam optimizer, a method that computes individual adaptive learning rates for different parameters using estimates of first and second moments of the gradients.

After the training, the model obtains the optimal weights and biases, those which minimize the loss function. These parameters are now ready to be used for prediction purposes.

4.2 VADER Sentiment analysis

Sentiment Analysis (also known as opinion mining) is a field of NLP that aims to identify, extract and quantify affective states and subjective information. In today's world, it is estimated that 80% of the data available is unstructured and not organized in any defined manner². Most of it comes in the form of text data like emails, chats, social media, articles, surveys, customer reviews, etc. Contrary to traditional tabular data, the analysis of texts is usually more complex, time consuming and expensive. In this sense, sentiment analysis systems allow a statistical approach to the study of text data, trying to make sense of all this unstructured information and get actionable insights in an efficient manner.

In recent years, specially with the raise of social media, sentiment analysis has become a topic of great interest. Reviews, ratings and other forms of recommendation, are a kind of virtual gold highly demanded by businesses looking to market their products more effectively, identify opportunities in the marketplace and manage their reputation. As an example, with the help of sentiment analysis, facebook posts can be automatically transformed into quantitative structured data from which is possible to infer public opinions about politics, sports, a particular company/service, or any topic people express their opinion about.

Furthermore, opinion mining algorithms can be applied to different levels of text like documents, paragraphs, sentences or even words within a sentence. Alike other NLP problems, sentiment analysis can be interpreted as a classification problem with different subproblems associated with attributes of the expression under study. The most basic task concerns the polarity of a given text level, classifying it as expressing

²The biggest data challenges that you might not even know you have, Cristie Schneider, May 25 (2016).

a positive, negative or neutral opinion. More advanced methods go beyond mere polarity classification and are capable of detecting feelings and emotions or identify intentions. Advantages of sentiment analysis include scalability which allows data processing at scale in a cost-effective way and, as a result, the chance of using the methodology during specific scenarios in real-time, identifying, and even anticipating, changes in opinions.

Methods and algorithms applied in sentiment analysis are primarily classified as:

- **Rule-based:** system which perform the analysis on the basis of a set of manually crafted rules.
- **Automatic:** sentiment analysis systems relying on machine learning algorithms that learn from data.
- **Hybrid:** systems combining the two previous types.

The first collection of methods is the only one employed in this study. Apparently, rule-based implementations seem to be quite simple; they can be applied by just defining two lists of polarized words, count the number of words from each list in a given text and if positives are higher output a 'positive' label, otherwise a 'negative'. Although theoretically valid, previous approach is very naïve since it does not consider the relative position of words in a sentence, potential subjectivity issues and boosting effects of certain words. For instance, if the number of positives exceeding negatives is relatively small and negatives are preceded by a word such as 'incredibly', it might be adventurous to label the sentence as 'positive'.

To deal with previously mentioned issues, among other aspects, this paper employs VADER³, a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media but proven to be quite successful when dealing with editorials and reviews. VADER uses a pre-built list of lexical features (e.g. words) which are labelled according to their semantic tendency as either positive or negative⁴. The main advantage of vader is that, instead of using discrete variables, it returns a numerical output capturing how positive or negative a sentiment is and the user can use this figure to create further labels. In addition, it does not require any training data as it is built from a generalizable, valence-based, human-curated gold standard sentiment lexicon (Hutto and Gilbert, 2014).

VADER is a fully open source developed under the MIT licence and the source code is available on github, together with a complete description of implementation details (see [VADER's github page](#) for further details). Among all VADER utilities, the following are noteworthy:

- **Punctuation:** The use of certain figures, such as exclamation marks (!), increase the magnitude of the intensity without modifying the semantic orientation. Appendix B contains the set of punctuation symbols that VADER considers to increase or decrease valences accordingly.
- **Capitalization:** All upper case letters to emphasize a sentiment-relevant word in the presence of other non-capitalized words, increases the magnitude of the sentiment intensity, again without changing semantic orientation.

³ VADER: Valence Aware Dictionary and sEntiment Reasoner

⁴Link to VADER's lexicon: [vader_lexicon.txt](#).

- **Boosters:** Booster dictionary listed in Appendix B includes intensifiers which impact the sentiment intensity by either increasing or decreasing the intensity. For instance words such as extremely, absolutely or highly have a positive impact on sentiment scores while partly, occasionally or little contribute negatively.
- **Conjunctions:** Words expressing contrast like 'but' suggest a shift in sentiment polarity, with the sentiment of the text following the conjunction being dominant. For instance, a sentence such as: "The place was good but, the weather was horrible" will be classified as negative.
- **Preceding tri-gram:** It is believed that by examining the word preceding a sentiment-laden lexical feature, nearly 90% of the negations are captured. VADER negate list displayed in Appendix B includes words that very likely flip the polarity of a text.

The code for applying VADER sentiment analysis to a text variable is more than simple. All the process described above, with all utilities, can be implemented in merely 8 lines of code by means of polarity_scores method (see listing 4.1).

```

1 import pandas as pd
2 from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
3
4 sia = SIA()
5 results = []
6
7 for line in news['Text2']:
8     pol_score = sia.polarity_scores(line)
9     pol_score['headline'] = line
10    results.append(pol_score)
11
12 df = pd.DataFrame.from_records(results)
13 df.head()
```

LISTING (4.1) Sentiment Intensity Analyzer

VADER sentiment analysis is applied to variable 'Text2' in the Reuters news dataset. Recall section 3.2 in chapter 3 describes the variables and, in this case, the one considered is the full text of the new, not the top story. When working with a massive number of news, it is more common to use headlines in sentiment analysis for time-saving and efficiency issues. However, the number of news retrieved for the three companies under study is handy enough to conduct the analysis on the full text instead of the top story. The justification behind the use of 'Text2' relies on the idea that complete texts might provide a more detailed description of the facts, hence the sentiment intensity analyzer will have more inputs to classify the instance. Nevertheless, both options should not be significantly different. Figure 4.1 displays the correlation table between negative, neutral and positive scores obtained after running listing 4.1 separately on variables 'Text1' and 'Text2'. Neg, neu and pos are, respectively, negative, neutral and positive valences for the headlines ('Text1'), and those followed by a 2 are the same measures for normal news ('Text2').

As expected, negative, neutral and positive scores are positively correlated and grouped together in the same cluster, as final results should be similar, no matter which 'Text' variable considered. In addition to scores for polarities above, the VADER sentiment intensity analyzer also provides a compound score, which is a synthetic measure that calculates the sum of all the lexicon ratings after being normalized between -1 (most extreme negative) and +1 (most extreme positive).

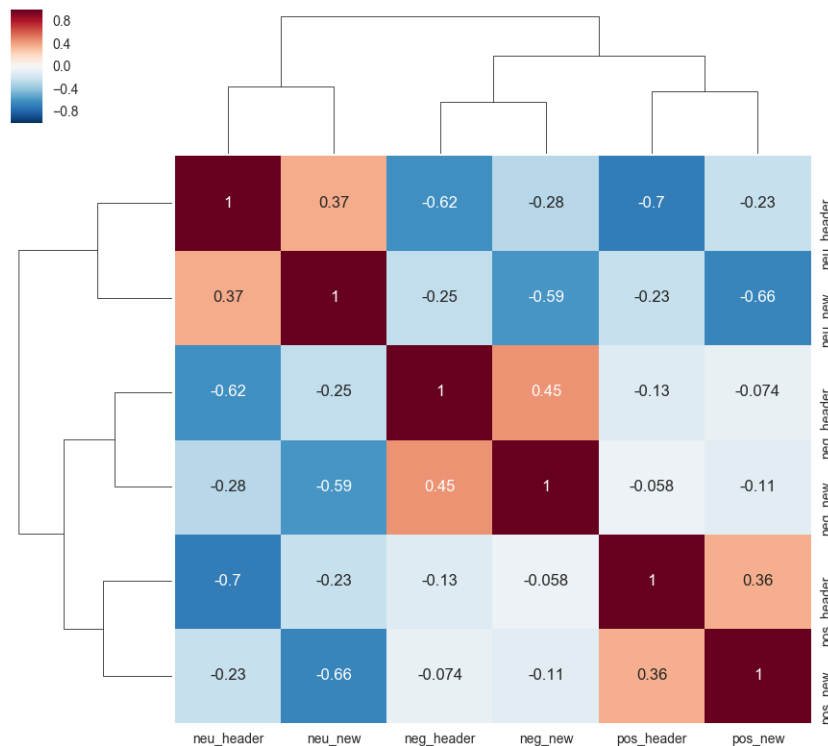


FIGURE (4.1) Correlation between headlines and normal news sentiment scores

Compound scores are the measure used to feed the network in the modelling stage. They represent a comprehensive score of polarities and are probably the most intuitive way to summarize overall sentiment.

News EDA in chapter 3 makes a brief mention to tokenization and feature engineering. Although beyond the scope of this study, since the main utility of these methods is for classification problems, once polarity scores has been obtained it is possible to define labels based on a certain criteria (e.g. 'positive' for scores ≥ 0.2 , 'negative' for scores ≤ -0.2 and 'neutral otherwise). Then, after removing 'stop words', the text can be splitted into senteces or words to further build some feature measures/figures such as most common words among texts labelled as 'positive', distribution of positive and negative word counts, etc.

This type of analysis is commonly referred to as feature extraction or feature engineering and has some relevance in stock market prediction problems, specially from a classification perspective. As this paper pursues to integrate news' sentiment in a regression problem via polarity scores, there is no need to conduct any further analysis beyond the application of VADER's Sentiment Intensity Analyzer.

4.3 LSTM for time series prediction

One of the fundamental problems which plagued traditional feed forward neural networks for a long time since its inception was the ability to identify patterns in sequence of inputs relying on previous sequences for information and context. From a theoretical point of view, taking into account that neural networks mimic biological neurons, it is totally justifiable to include the concept of memory in artificial NN systems since humans do not start their learning process from scratch every second. For instance, when reading a text, the meaning of words is based on the understanding of previous words in a sentence which allow for a context to predict what the next word might be.

Traditional neural networks cannot do this, they take in a stand-alone data vector of some dimension each time and past information do not persist. A few years ago, this lack of memory seemed a major shortcoming in the learning process since it was unclear how traditional neural networks could use information about previous events to predict later ones.

First attempts to tackle this issue used feed forward neural networks with the peculiarity that the output was fed-back as a new input to provide context on the last seen events. These architectures are no more than traditional networks with loops inside them, allowing information to persist. As an example 4.11 presents a simple recurrent neural as a length-2 sequence with tanh activations:

$$\begin{aligned}
 \hat{y}_t &= f(y_{t-1}, y_{t-2}) \\
 \hat{y}_t &= h_t V + c \\
 h_t &= \tanh(h_{t-1} W + y_{t-1} U + b) \\
 h_{t-1} &= \tanh(h_{t-2} W + y_{t-2} U + b) \\
 h_{t-2} &= 0
 \end{aligned} \tag{4.11}$$

V is the output of hidden layer t which includes weights W and data units U from $t - 1$. In fact, RNN are multiple copies of the same network, each one passing a message to subsequent t . While these architectures worked to an extent, the idea that they are capable of relating previous information to the present event is not always true. RNN specially suffer from 'long-term' dependencies, situations where the time gap between the present and the past relevant information becomes large⁵. This represents a major downfall since, in practice, many time series modelling problems require the information enclosed in past distant data for future predictions.

In theory, RNN are absolutely capable of dealing with such 'long-term dependencies' but the mechanics suffer from the so called vanishing gradient problem when trying to learn from distant events. In accordance with the structure defined in ??, a RNN with sequence length x can be 'unrolled' so that each step back in the past is added to the model as a new hidden layer. For instance, previous example is essentially a two hidden layer network where t considers information exclusively from the two previous time moments. As more layers are added to the network, to capture more distant observations, certain activation functions return gradient values close to zero, making the network hard to train. Figure 4.2 depicts an unrolled RNN for some t sequence length being A neural network chunks.

⁵(Bengio, Simard, and Frasconi, 1994) explores the problem in detail.

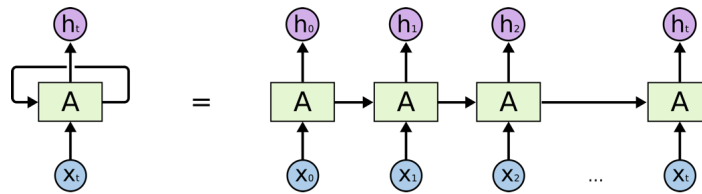


FIGURE (4.2) An unrolled RNN

Source: Colah's blog

Recall the gradients are found using backpropagation and the chain rule, in simple terms, multiplying the values of each layer's derivatives down the network. Dense functions that squish a large input space into a small one such as sigmoid, tanh or ISRU; are very robust to significant changes in the input hence derivatives become small and, when multiplied together, these small values exponentially decrease the gradient as propagated down to the initial layers. Possible remedies to the vanishing gradient problem include identity RNN (same architecture using ReLU activations), batch normalization or even gradient clipping.

Despite previous solutions achieved certain success, RNN were still poorly suited in most real-world problems, thus other ways of tackling the long term memory issue needed to be found. Thereupon, in 1997, Hochreiter and Schmidhuber presented their Long Short Term Memory networks, the first architecture capable of learning long-term dependencies (Ferrara et al., 2014). It was a real phenomenon in the field as their methodology worked tremendously well on a wide variety of problems and still the most popular method nowadays for time series modelling.

LSTM model are explicitly designed to resolve the long term dependency problem and remembering distant events is practically their default behaviour. These models work, for many tasks, much better than standard versions of RNN, outperforming them in practically all scenarios and getting rid of the vanishing gradient problem.

Differences between standard RNN and LSTM models are noticeable, although the two architectures present the characteristic chain like structure discussed earlier. Nevertheless, while the repeating module in standard RNN has a very simple structure (in most of the cases just a single layer), LSTM is built with a repeating module containing four interacting layers. Figure 4.3 depicts a diagram of a LSTM architecture, being the yellow rectangles the neural network layers inside the repeating module ⁶.

The main key to success of LSTM models is the inclusion of a cell state in the network flow (the horizontal line at the top of the diagram). Cell states are very important objects in LSTMs since they have few linear interactions and it is very easy for them to transmit information through the different network layers practically unchanged, hence keeping the track of past distant events.

Recall *As* represent neural network pieces thus, in the LSTM case, each repeating module can be thought of as a neural network inside another neural network. The core idea, and also the main innovation behind LSTMs is that the model has the ability to remove or add information to the cell state based on its relevance.

⁶ A standard RNN will only have one layer of the corresponding activation function, and the network's output will be transferred directly to the next *A* chunk.

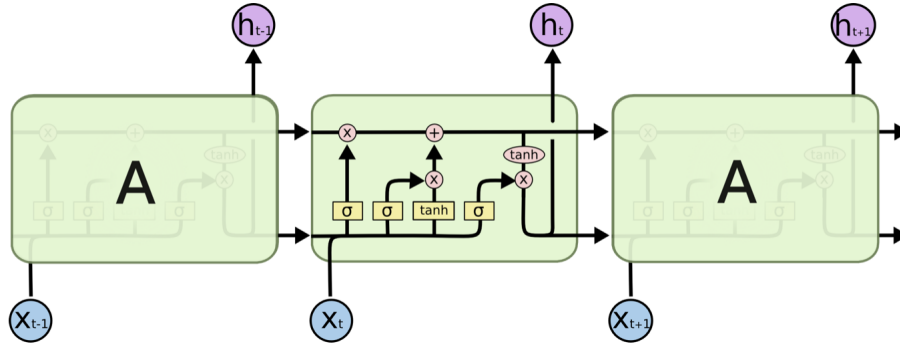


FIGURE (4.3) LSTM architecture diagram
Source: Colah's blog

This process is regulated by means of sigmoid layers which describe how much of the output should be retained and transferred to further time moments or intervals. These sigmoid layers are similar to 'gates' regulating which previous information should be kept and which should be dismissed.

Each A chunk in figure 4.3 takes as input previous layer cell state C_{t-1} , together with h_{t-1} and data points corresponding to time t , X_t .

Following the process flow described in figure 4.3, first step in an LSTM is to decide which past information should be discarded from the cell state. The object responsible for this task is the first sigmoid layer called the 'forget gate layer'. Using Colah's notation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (4.12)$$

The output of a sigmoid layer is a number between 0 and 1 and the operation is performed for each number in the cell state C_{t-1} . A value of 1 means to keep everything while 0 stands for dismissing all the information. 'Forget gate layers' act as preliminary filters to maintain only relevant information in cell states and avoid the transfer of noisy or trivial data to further layers.

Next step is to update previous parameters, on the basis of which new information is going to be stored in the cell state. This process has two parts: first the second sigmoid decides which cell values to update and next (input gate layer), a tanh layer (or an equivalent activation) creates a vector of new candidates that might be potentially added to cell state C_{t-1} . Again, using Figure 4.3 notation:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \quad (4.13)$$

These two objects are further combined and added to the product of previous cell states and the output of the previously calculated 'forget gate layer'. Therefore, the new cell state C_t will be:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (4.14)$$

The new state is calculated with contributions from previous states and new information of time t , in both cases previously filtered by applying a sigmoid layer to capture only relevant information.

Finally, last sigmoid layer deduces what to output based on previously calculated cell state C_t . After applying the sigmoid for filtering purposes, the cell state is passed to a tanh activation (output between -1 and 1) and the result is multiplied by the output of the sigmoid gate. Once again, in Colah's notation:

$$\begin{aligned}\sigma_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= \sigma_t \cdot \tanh(C_t)\end{aligned}\tag{4.15}$$

Equation 4.15 is the output of a regular LSTM layer. Previously described architecture represents a normal LSTM but, although the core idea of the four gates is essentially the same, the actual placement of the arrows differs depending on the research paper allowing for slightly different versions of the model. Differences are minor, mainly related to interactions between the four gates, looking forward to simplify the model and adapt it better to the problem treated.

In this paper, the LSTM model employed was originally designed by Jakob Aungiers, 2018 and further adapted to include sensitivity scores. As discussed in the beginning of the chapter, the motion of a stock market time series is not any sort of known function, in fact, the best property to describe them appears to be a random walk. In theory, a random walk has no predictable patterns hence attempting to model will be worthless but, fortunately, there are on-going arguments by many experts in the field arguing that stock price movements are not pure random walks and they have some sort of hidden pattern.

An important remark is that, as stock prices are constantly moving with no specified boundaries, data normalization is needed, otherwise the model will never converge. To combat this, a recommended procedure is to normalize the data by taking percentage changes from the start of the window hence, in the case of financial time series, the output feeded to the network will reflect returns on stock over a rolling window of width equal to the specified sequence length. Following equation is used to normalize the data:

$$n_i = \left(\frac{p_i}{p_0} \right) - 1\tag{4.16}$$

Chapter 5

Results

After the merging process, a variable with compound scores is added to the stock prices dataset, reflecting the news' sentiment of those dates with news available. The model is programmed to deploy those observations with NAs as it will not be possible to calculate the loss function otherwise. Whilst this work aims to exhibit a practical example of LSTM neural networks, it only scratches the surface of the tremendous potential the method has, not only in the financial sector but also in any time series prediction problem.

There is still room for improvement in all models presented and greater accuracy results could be certainly achieved with more iterations on deeper networks and proper hyperparameter tuning. Subsequent architectures are, to some extent, naïve models which pretend to open the doors for further reaseach in a field with huge potential.

The LSTM methodology is implemented by means of a run file which calls methods inside data processor and model class objects defined in Appendix A. These python classes require as arguments to their methods a network design together with some hyperparameters, dimensionality of the inputs and some other specifications. All these figures are provided to the network from a json file which contains the model settings and configuration. This approach is genuinely worthwhile owing to its flexibility, allowing the development of highly customizable models by only changing the settings and runing the script again.

By way of example, listing 5.1 presents one of the configurations used in this study.

```

1 {
2   "data": {
3     "filename": "MSFT_scores.csv",
4     "columns": [
5       "Adj Close",
6       "compound"
7     ],
8     "sequence_length": 21,
9     "train_test_split": 0.85,
10    "normalise": true
11  },
12  "training": {
13    "epochs": 2,
14    "batch_size": 32
15  },
16  "model": {
17    "loss": "mse",
18    "optimizer": "adam",
19    "save_dir": "saved_models",
20    "layers": [
21      {
22        "type": "lstm",

```

```
23     "neurons": 200,
24     "input_timesteps": 20,
25     "input_dim": 2,
26     "return_seq": true
27   },
28   {
29     "type": "dropout",
30     "rate": 0.2
31   },
32   {
33     "type": "lstm",
34     "neurons": 200,
35     "return_seq": true
36   },
37   {
38     "type": "lstm",
39     "neurons": 200,
40     "return_seq": false
41   },
42   {
43     "type": "dropout",
44     "rate": 0.2
45   },
46   {
47     "type": "dense",
48     "neurons": 1,
49     "activation": "linear"
50   }
51 ]
52 }
53 }
```

LISTING (5.1) LSTM settings

JSON formats resemble a combination of python lists and dictionaries, thus it is very easy for python to parse it and access the attributes. Previous configuration includes specifications about the data to use, the training process and the model architecture.

Notice the model uses only 2 training epochs and a reasonable batch size to speed up the training process. Dropout is a commonly used regularization technique for reducing overfitting. It basically consists in cutting some connections between both hidden and visible neurons. Sequence length considered is 21 days, a period approximately equivalent to a natural month.

Next steps implement the model on company price datasets, enriched with compound scores.

5.1 Unidimensional LSTM prediction

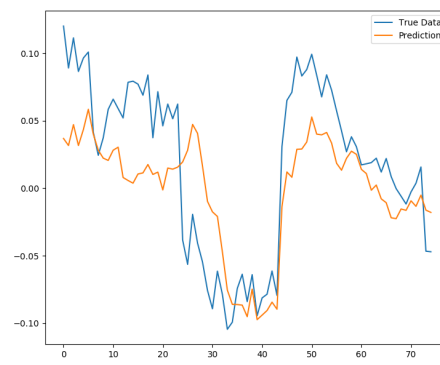
In the first instance, unidimensional models are developed hence the only column that should appear in the settings file is 'Adj Close' and the input dimension has to be resetted to 1. The model's design is the following:

- **Layers:** (1, 200, 200, 200, 1).
 - *LSTM*: 200 neurons \rightarrow (1, 200).
 - *LSTM*: 200 neurons \rightarrow (200, 200).
 - *Linear*: 1 fully connected output \rightarrow (200, 1).

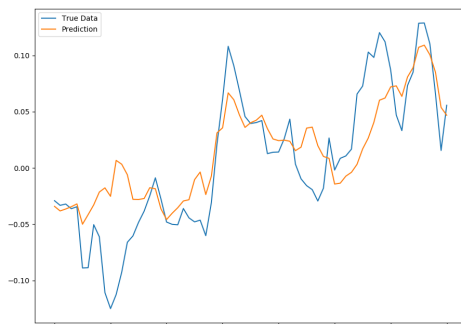
- **Epochs:** 2.
- **Batch Size:** 32.
- **Train-test split:** 85%.
- **Loss function:** MSE.
- **Optimizer:** Adam.
- **Sequence length:** 21.



(A) MSFT



(B) JNJ



(C) JPM

FIGURE (5.1) Unidimensional point-by-point prediction

As its name suggests, point-by-point predictions shown in figure 5.1 are only predicting a single point ahead each time using data from previous 21 observations. Then, the program plots the point and moves towards the next training window to repeat the same process again.

In addition to the plots, results for the three companies are the following ones:

- **MSFT**
 - *Date range:* 06/07/2011 - 10/06/2019.
 - *Time periods:* 1.024.
 - *Training time:* 6,98 seconds.
 - *Training loss in last iteration:* 0,0015.
- **JNJ**
 - *Date range:* 07/07/2011 - 06/06/2019.

- *Time periods*: 638.
- *Training time*: 48,29 seconds.
- *Training loss in last iteration*: 0,0024.

- **JPM**

- *Date range*: 17/02/2015 - 10/06/2019.
- *Time periods*: 609.
- *Training time*: 5,35 seconds.
- *Training loss in last iteration*: 0,0016.

Predictions in figure 5.1 seem to match returns quite accurately. However, previous exercise is of little practical utility since the network is predicting returns one step ahead the prior true history window. In reality, the model is just answering the question what is going to happen tomorrow.

Furthermore, even if a prediction for time $t + 1$ is incorrect, next window will automatically deploy this value replacing it with the true result after reaching $t + 1$ (which will become the new t). In consequence, previous models are not capable of predicting future returns over a relatively lengthy period, although still useful in volatility forecasting as they provide a representation of the ranges that prices should be moving within.

In order to deal with previous concerns and to provide a model of greater utility in a life trading system, multi-sequence prediction is a commonly used technique to compute LSTM predictions for X -steps ahead.

The mechanics are quite simple, the model also initializes the test window with test data and predicts the next point. Thus far, it is the same as a regular point-by-point prediction. However, instead of using test data in forthcoming windows, it uses the previous predicted point to calculate the next one, and so on. The process continues until the input window is fully made of past predictions. At such time, it shifts forward one full window length through the test data and resets the window to start the process again. In this case, each training point is feeded only once in the model.

The network architecture remains unchanged although a smaller batch size of 8 has been considered to reinforce the learning and also the sequence length has been shortened to 11 periods. The main justification behind this decision relies on the fact that data is significantly scarce in the current scenario hence excessively broad windows will prompt the model to work mainly with predicted points and forego relevant information in the testing dataset. Figure 5.2 shows the implementation of multi-sequence predictions.

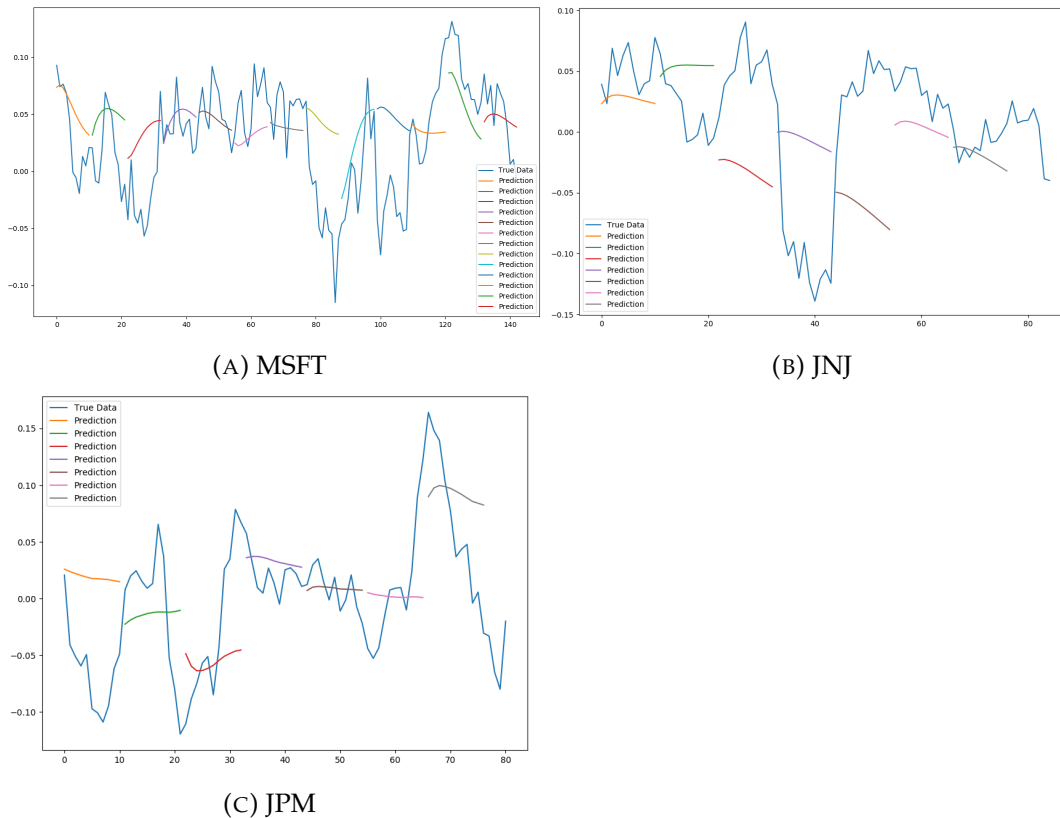


FIGURE (5.2) Unidimensional multi-sequence prediction

Results obtained are the following for the three companies:

- **MSFT**

- *Date range:* 06/07/2011 - 10/06/2019.
- *Time periods:* 1.024.
- *Training time:* 12,42 seconds.
- *Training loss in last iteration:* 8,4948e-04.

- **JNJ**

- *Date range:* 07/07/2011 - 06/06/2019.
- *Time periods:* 638.
- *Training time:* 58,57 seconds.
- *Training loss in last iteration:* 0,0011.

- **JPM**

- *Date range:* 17/02/2015 - 10/06/2019.
- *Time periods:* 609.
- *Training time:* 1 minute 56,61 seconds.
- *Training loss in last iteration:* 0,0013.

Whilst far from perfect, multiple-trend like predictions seek to capture future momentum trends which, translated to trading, represent the capacity of benefiting from future ups and downs in the market.

5.2 Multidimensional LSTM prediction

Thus far, previous models only consider a single variable in their analysis, concretely 'Adj Close' price. However, in this type of analysis, it is believed that other variables related to price data can enrich the training dataset, ultimately leading to a higher performance of the model.

Chapter 3 showed that Volume and news' sentiment are magnitudes certainly correlated with adjusted close prices therefore, if VADER polarity scores were to be accurate estimates of news' sentiment, both variables will presumably improve previously obtained result. Accordingly, these two variables are the 'enriching candidates' considered.

In this section, the objective is to determine whether Volume or compound scores are better enriching factors for company price data in terms of model performance. In line with the argument presented in previous section, multi-sequence predictions are the only ones considered from now on, since point-by-point has proven to be of doubtful practical utility in life trading systems.

LSTM class methods in Appendix A are capable of handling multiple dimensions and, this time, the settings file should be modified in the opposite way as before. Columns should include the two variables considered and the input dimension field has to be assigned a value of 2. Other settings are the same as in the unidimensional case.

- **Layers:** (1, 200, 200, 200, 1).
 - LSTM: 200 neurons \rightarrow (1, 200).
 - LSTM: 200 neurons \rightarrow (200, 200).
 - Linear: 1 fully connected output \rightarrow (200, 1).
- **Epochs:** 2.
- **Batch Size:** 8.
- **Train-test split:** 85%.
- **Loss function:** MSE.
- **Optimizer:** Adam.
- **Sequence length:** 11.

Predictions using compound scores are displayed in figure 5.3, together with the corresponding results for each firm:

- **MSFT**
 - Date range: 06/07/2011 - 10/06/2019.
 - Time periods: 1.024.
 - Training time: 18,62 seconds.
 - Training loss in last iteration: 0,0034.
- **JNJ**
 - Date range: 07/07/2011 - 06/06/2019.

- Time periods: 638.
- Training time: 11,46 seconds.
- Training loss in last iteration: 0,0024.

- JPM

- Date range: 17/02/2015 - 10/06/2019.
- Time periods: 609.
- Training time: 10,67 seconds.
- Training loss in last iteration: 0,0031.

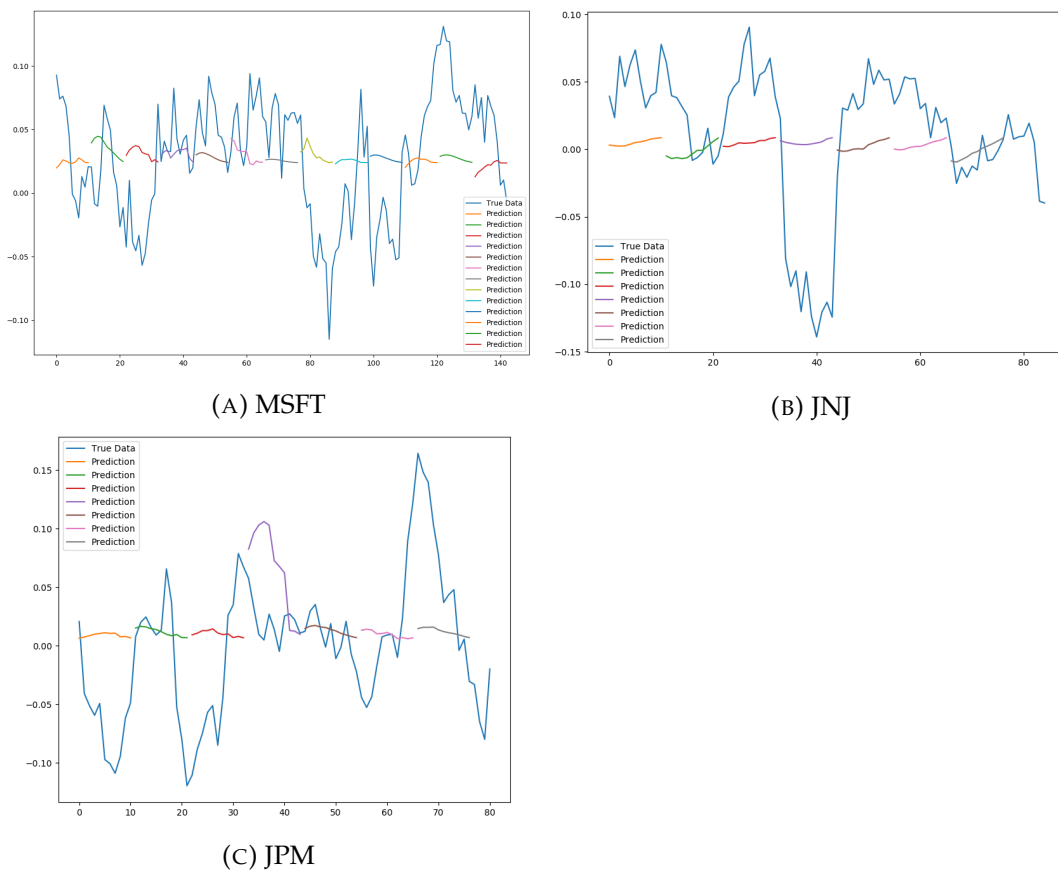


FIGURE (5.3) Multidimensional multi-sequence prediction (compound)

Similarly, figure 5.4 does the same, this time using 'volume' as the enriching factor.

- MSFT

- Date range: 06/07/2011 - 10/06/2019.
- Time periods: 1.024.
- Training time: 2 min 33,32 seconds.
- Training loss in last iteration: 0,0016.

- **JNJ**
 - Date range: 07/07/2011 - 06/06/2019.
 - Time periods: 638.
 - Training time: 10,84 seconds.
 - Training loss in last iteration: 0,0021.
- **JPM**
 - Date range: 17/02/2015 - 10/06/2019.
 - Time periods: 609.
 - Training time: 11,86 seconds.
 - Training loss in last iteration: 0,0024.

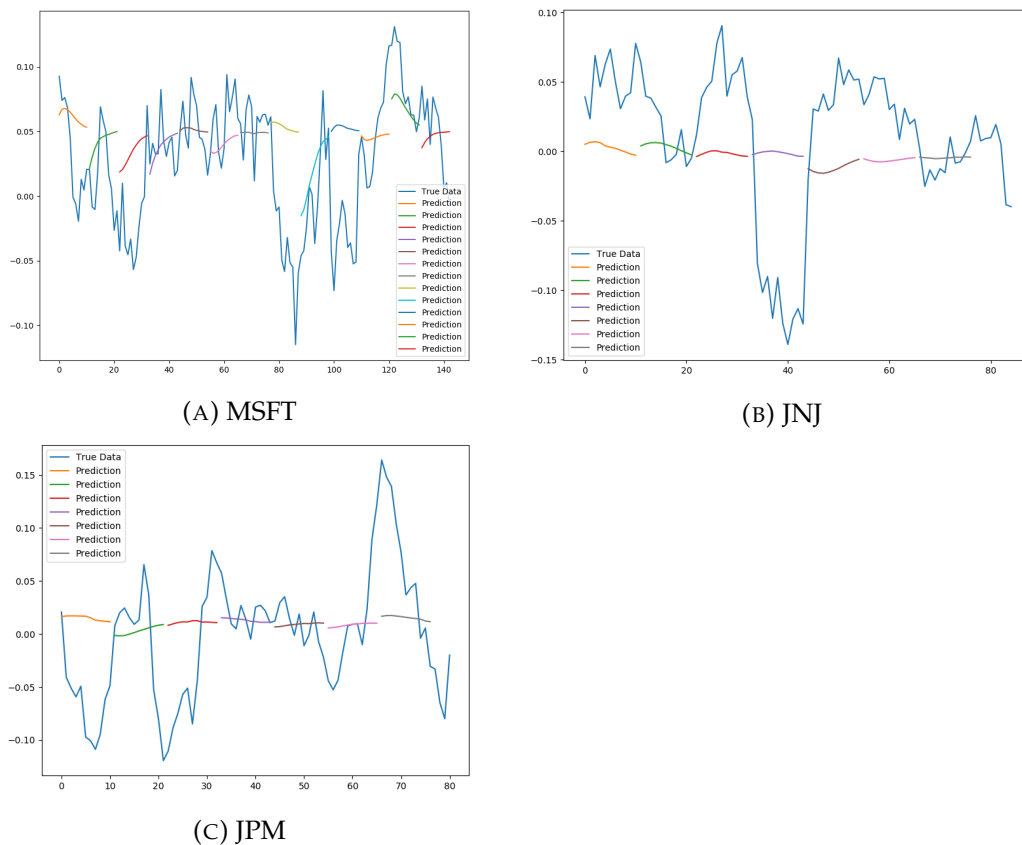


FIGURE (5.4) Multidimensional multi-sequence prediction (volume)

Eventually, although not a fair comparison, an investor may actually wonder why not using stock prices and volumes directly from files retrieved from Yahoo finance. Recall previous models consider only those observations with non-missing compound scores hence, if the focus is exclusively on volume, information from yahoo finance seems to be a better option since no dates will be missing and the obtention process is much faster.

Figure 5.5 displays a multi-sequence prediction of length 40 and a 24 batch size on full time series of company price data using 'volume' as the enriching factor.

Modifications on batch size and sequence length respond to the larger datasets considered in this case, seeking to speed up the learning process.

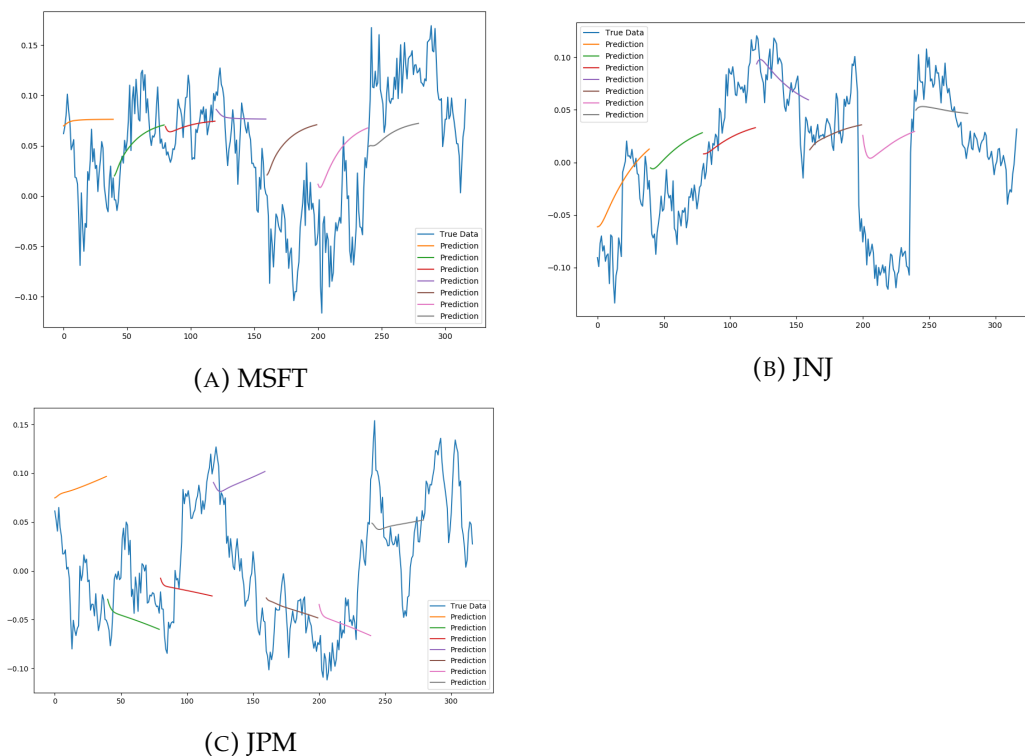


FIGURE (5.5) Multi-sequence prediction on complete stock data

Previous model results for the three companies under study are the following:

- **MSFT**

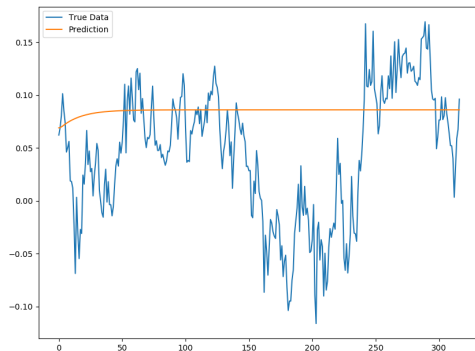
- *Date range:* 04/01/2010 - 10/06/2019.
- *Time periods:* 2.374.
- *Training time:* 3 min 00,49 seconds.
- *Training loss in last iteration:* 0,0018.

- **JNJ**

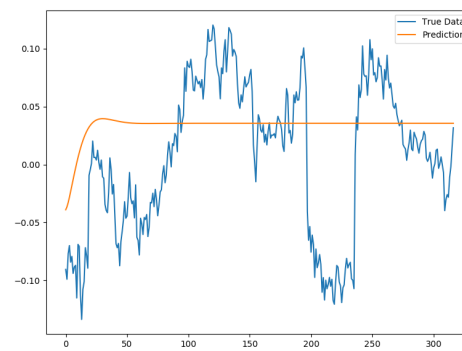
- *Date range:* 04/01/2010 - 10/06/2019.
- *Time periods:* 2.374.
- *Training time:* 33,58 seconds.
- *Training loss in last iteration:* 9,0235e-04.

- **JPM**

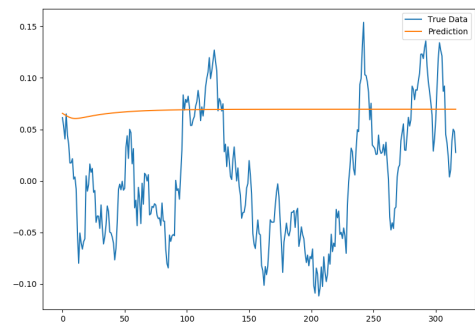
- *Date range:* 04/01/2010 - 10/06/2019.
- *Time periods:* 2.374.
- *Training time:* 2 mins 59,06 seconds.
- *Training loss in last iteration:* 0,0021.



(A) MSFT



(B) JNJ



(C) JPM

FIGURE (5.6) Full-sequence prediction on complete stock data

In general, multi-sequence predictions does appear to be correctly predicting the trend for a good majority of the time series.

Lastly, with an enough rich dataset, it is also possible to conduct a full sequence prediction. This procedure is similar to the multi-sequence case despite the fact that, this time, only the first testing window is initialized with test data, while the rest of the predictions are entirely based on previously predicted points (see figure 5.6).

Chapter 6

Concluding remarks

Today's world is being increasingly driven by data. Consumers, companies, public institutions and other agents throughout the globe are constantly generating massive amounts of data. Almost any day-to-day action leaves behind a data trace that can be further harvested, preprocessed and analyzed making up the basis of 'Big Data'.

Even though 'Big Data' is a complex issue, if treated appropriately, it can be turned into really valuable insights to inform competitive decision-making and generate value in businesses from a wide range of industries. However, such a boost in the scale of the information available and steadily generated requires an adaptation of the traditional data science process to more flexible methods capable of living in a highly dynamic environment.

Operational velocity is one of the main issues addressed in this new era, not only in data analysis but also throughout all stages of the data science process. More powerful and sophisticated technologies for data processing are in the market today, thus data collection techniques should be also aligned with these recent innovations, and web scraping methods are undoubtedly into the spotlight.

Traditional data collection systems are becoming of less practical utility as time goes by, being replaced with new mechanisms that essentially 'harvest' data instead of collecting in. With the irruption of Web 2.0, and in today's world of networked and distributed applications, the Internet is becoming the quintessentially data source as practically everything is shared through it and, consequently, it contains a real universe of information awaiting for a retrieval.

Web scraping tools allow researchers and businesses to monitor information over the net and also an automated data collection on a routinary basis. They also permit a highly customizable approach to data extraction in an efficient and fast way, visiting a large number of webpages simultaneously. The vast field of web scraping applications is truly mind-boggling as information on the net is extremely heterogenous. Competitor price monitoring, market data, financial statements or movie reviews are just a few examples of data that scrapers can harvest over the net, to thereafter use them in real-time analytics, as the information can be analyzed right after becoming available.

A web based approach to data extraction processes can definitely improve the quality of our data products. This added value is certainly noticed in the application of such methods described in this paper, as financial markets are a clear example of an scenario where the abovementioned operational velocity is no longer a desire but a necessity.

Either by using an API or by extracting html references from webpages, the methodology employed enables an instantaneous disposal of highly heterogenous data in large quantities. Furthermore, pre-built datasets of the information demanded

for this work, with an special emphasis on news data, are either scarce or non-existent, hence web scraping is, in fact, the only plausible alternative to effectively retrieve the data needed.

Additionally, a deep learning framework for predictive modelling is also developed, aiming to illustrate a possible use of the already retrieved data in line with previously mentioned principles of large data quantities and real-time analysis. A standard LSTM neural network model is considered in the modelling stage as it has proven to be a very consistent method for characterizing time series data. The hypothesis suggested pretends to test if news data can be a 'enriching factor' to stock price predictions, thus improving the accuracy of our models.

Before delving deeper into the details, an important consideration is that all models displayed in chapter 5 can achieve greater accuracy with more training and careful hyperparameter tuning. Having said that, figures 5.1a, 5.1b and 5.1c evince the model is predicting the series very accurately on a point-by-point basis without any enriching factor. As discussed earlier, although of little relevance in a life trading system, these predictions are still useful in certain applications such as volatility analysis or anomaly detection.

As moving towards, in multi-sequence prediction, overfitting is the dominant note in practically all scenarios, presumably due to data scarcity issues. Training losses are small but still the trained network fails to capture certain trends when tested against the 'ground truth'. Overfitting is not resolved by just increasing the epochs in the network, it is a concern which requires further research. A good start, perhaps, is to study the distribution of training losses against test losses.

Besides overfitting issues, if models in figures 5.3 and 5.4 are compared, the inclusion of polarity scores from news data seems to generate smoother predictions although in many cases, trend estimates are better. On the contrary, enriching prices with volume makes the model more sensitive to abrupt price variations, although if lacking them, the model predominantly fails in estimating medium term trends for those stocks which are generally more stable. In addition, enriching factors such as compound polarity scores or volume partially resolve the data scarcity problem since the network is feeded with more information that can be considered in predictions (compare figures ?? and ?? vs, for instance, figures ?? and ??). In general, although long-term trends seem to be better predicted by including polarity scores as enrichers instead of volumes, there is not enough evidence to either accept or reject the previously stated hypothesis since more data has to be feeded into the model and, with the available computing resources, the shown data volume is the maximum plausible achievement.

Nevertheless, in terms of practical utility and real applications of the methodology it is preferable to opt for volumes as the data retrieval process is easier, and prediction results achieved when working with complete stock data are certainly successful. In the three cases (figure 5.5c) trend estimates seem to correctly capture future data tendencies and, in the case of Johnson & Johnson (figure ??) the model is also capable of outputting a sensible estimate of the trend over the entire time horizon considered. Notice how meaningful is this result since we are saying that given 40 days of data entries, the LSTM model can predict a trend for the next 2300 days (approx.) using exclusively the previously predicted values as inputs.

Appendix A

Python class objects

A.1 Reuters news crawler

```

1 import os
2 import sys
3 import datetime
4
5 # import News data scraping utilities from parent directory
6 import inspect
7 currentdir = os.path.dirname(os.path.abspath(inspect.getfile(inspect.
    currentframe()))))
8 parentdir = os.path.dirname(os.path.dirname(currentdir))
9 sys.path.insert(0, parentdir)
10 import utilities
11
12
13 class ReutersCrawler(object):
14
15     def __init__(self):
16         self.ticker_list_filename = './input/tickerList.csv'
17         self.finished_reuters_filename = './input/finished_reuters'
18         self.failed_reuters_filename = './input/news_failed_tickers.csv'
19         self.news_filename = './input/news_reuters.csv'
20
21     def calc_finished_ticker(self):
22         # call this funtion when restarting a task
23         os.system("awk -F',' '{print $1}' ./input/news_reuters.csv | sort
    | uniq > ./input/finished_reuters")
24
25     def load_finished_tickers(self):
26         # Load the already finished reuters if any
27         return set(self._load_from_file(self.finished_reuters_filename))
28
29     def load_failed_tickers(self):
30         failed_tickers = {} # {ticker: priority}
31         for line in self._load_from_file(self.failed_reuters_filename):
32             ticker, _, priority = line.split(',')
33             failed_tickers[ticker] = priority
34         return failed_tickers
35
36     def _load_from_file(self, filename):
37         if os.path.exists(filename):
38             with open(filename, 'r') as f:
39                 for line in f:
40                     yield line.strip()
41
42     def fetch_content(self, task, date_range):
43
44         ticker, name, exchange, market_cap = task

```

```

45     print("%s - %s - %s - %s" % (ticker, name, exchange, market_cap))
46
47     suffix = {'AMEX': '.A', 'NASDAQ': '.O', 'NYSE': '.N'}
48     link = "https://www.reuters.com/finance/stocks/company-news/"
49     url = link + ticker + suffix[exchange]
50
51     ticker_failed = open(self.failed_reuters_filename, 'a+')
52     today = datetime.datetime.today().strftime("%Y%m%d")
53
54     news_num = self.get_news_num_whenever(url)
55     if news_num:
56         # If news, fetch for N consecutive days in the past
57         has_content, no_news_days = self.fetch_within_date_range(
news_num, url, date_range, task, ticker)
58         if not has_content:
59             print('%s has no content within date range' % ticker)
60         if no_news_days:
61             print('set as LOW priority')
62             for timestamp in no_news_days:
63                 ticker_failed.write(ticker + ',' + timestamp + ',' + '
LOW\n')
64         else:
65             # The company has no news even with unspecified date
66             # add it into the lowest priority list
67             print("%s has no news at all, set as LOWEST priority" % (
ticker))
68             ticker_failed.write(ticker + ',' + today + ',' + 'LOWEST\n')
69             ticker_failed.close()
70
71     def get_news_num_whenever(self, url):
72         # check if the ticker has any news
73         # return the number of news
74         soup = utilities.get_soup_with_repeat(url, repeat_times=4)
75         if soup:
76             return len(soup.find_all("div", {'class': ['topStory', '
feature']}))
77         return 0
78
79     def fetch_within_date_range(self, news_num, url, date_range, task,
ticker):
80         # No news for X consecutive days, stop iterating dates
81         # Second-lowest priority list
82         missing_days = 0
83         has_content = False
84         no_news_days = []
85         for timestamp in date_range:
86             print('trying '+timestamp, end='\r', flush=True)
87             new_time = timestamp[4:] + timestamp[:4]
88             soup = utilities.get_soup_with_repeat(url + "?date=" +
new_time)
89             if soup and self.parse_and_save_news(soup, task, ticker,
timestamp):
90                 missing_days = 0 # if get news, reset missing_days as 0
91                 has_content = True
92             else:
93                 missing_days += 1
94
95             # the more news_num, the longer we can wait
96             if missing_days > news_num * 5 + 20:
97                 # no news in X consecutive days, stop crawling
98                 print("%s has no news for %d days, stop this candidate ...
" % (ticker, missing_days))
99                 break

```

```

100         if missing_days > 0 and missing_days % 20 == 0:
101             no_news_days.append(timestamp)
102
103         return has_content, no_news_days
104
105     def parse_and_save_news(self, soup, task, ticker, timestamp):
106         content = soup.find_all("div", {'class': ['topStory', 'feature']})
107         if not content:
108             return False
109         with open(self.news_filename, 'a+', newline='\n') as fout:
110             for i in range(len(content)):
111                 title = content[i].h2.get_text().replace(", ", " ").replace(
112 ("\\n", " "))
113                 body = content[i].p.get_text().replace(", ", " ").replace(
114 "\\n", " ")
115
116                 if i == 0 and soup.find_all("div", class_="topStory"):
117                     news_type = 'topStory'
118                 else:
119                     news_type = 'normal'
120
121                 print(ticker, timestamp, title, news_type)
122                 fout.write(','.join([ticker, task[1], timestamp, title,
123 body, news_type])+ '\\n')
124
125         return True
126
127     def run(self, numdays=1000):
128         """Start crawler back to numdays"""
129         finished_tickers = self.load_finished_tickers()
130         failed_tickers = self.load_failed_tickers()
131         date_range = utilities.generate_past_n_days(numdays)
132
133         # store low-priority task and run later
134         delayed_tasks = {'LOWEST': set(), 'LOW': set()}
135         with open(self.ticker_list_filename) as ticker_list:
136             for line in ticker_list:
137                 task = tuple(line.strip().split(','))
138                 ticker, name, exchange, market_cap = task
139                 if ticker in finished_tickers:
140                     continue
141                 if ticker in failed_tickers:
142                     priority = failed_tickers[ticker]
143                     delayed_tasks[priority].add(task)
144                     continue
145                 self.fetch_content(task, date_range)
146
147         # run low priority
148         for task in delayed_tasks['LOW']:
149             self.fetch_content(task, date_range)
150
151         # run lowest priority
152         for task in delayed_tasks['LOWEST']:
153             self.fetch_content(task, date_range)

```

LISTING (A.1) Reuters crawler class object

A.2 LSTM data processor

```

1 import math
2 import numpy as np
3 import pandas as pd
4
5 class DataLoader():
6     """A class for loading and transforming data for the lstm model"""
7
8     def __init__(self, filename, split, cols):
9         dataframe = pd.read_csv(filename)
10        i_split = int(len(dataframe) * split)
11        self.data_train = dataframe.get(cols).values[:i_split]
12        self.data_test = dataframe.get(cols).values[i_split:]
13        self.len_train = len(self.data_train)
14        self.len_test = len(self.data_test)
15        self.len_train_windows = None
16
17    def get_test_data(self, seq_len, normalise):
18        """
19        Create x, y test data windows
20        """
21        data_windows = []
22        for i in range(self.len_test - seq_len):
23            data_windows.append(self.data_test[i:i+seq_len])
24
25        data_windows = np.array(data_windows).astype(float)
26        data_windows = self.normalise_windows(data_windows, single_window=
False) if normalise else data_windows
27
28        x = data_windows[:, :-1]
29        y = data_windows[:, -1, [0]]
30        return x,y
31
32    def get_train_data(self, seq_len, normalise):
33        """
34        Create x, y train data windows
35        """
36        data_x = []
37        data_y = []
38        for i in range(self.len_train - seq_len):
39            x, y = self._next_window(i, seq_len, normalise)
40            data_x.append(x)
41            data_y.append(y)
42        return np.array(data_x), np.array(data_y)
43
44    def generate_train_batch(self, seq_len, batch_size, normalise):
45        """Yield a generator of training data from filename on given list
of cols"""
46        i = 0
47        while i < (self.len_train - seq_len):
48            x_batch = []
49            y_batch = []
50            for b in range(batch_size):
51                if i >= (self.len_train - seq_len):
52                    # stop-condition for a smaller final batch if data
doesn't divide evenly
53                    yield np.array(x_batch), np.array(y_batch)
54                    i = 0
55                x, y = self._next_window(i, seq_len, normalise)
56                x_batch.append(x)
57                y_batch.append(y)

```

```
58         i += 1
59         yield np.array(x_batch), np.array(y_batch)
60
61     def _next_window(self, i, seq_len, normalise):
62         '''Generates the next data window from the given index location i
63         '''
64         window = self.data_train[i:i+seq_len]
65         window = self.normalise_windows(window, single_window=True)[0] if
normalise else window
66         x = window[:-1]
67         y = window[-1, [0]]
68         return x, y
69
70     def normalise_windows(self, window_data, single_window=False):
71         '''Normalise window with a base value of zero'''
72         normalised_data = []
73         window_data = [window_data] if single_window else window_data
74         for window in window_data:
75             normalised_window = []
76             for col_i in range(window.shape[1]):
77                 normalised_col = [((float(p) / float(window[0, col_i])) -
1) for p in window[:, col_i]]
78                 normalised_window.append(normalised_col)
79                 normalised_window = np.array(normalised_window).T # reshape
and transpose array back into original multidimensional format
80             normalised_data.append(normalised_window)
81         return np.array(normalised_data)
```

LISTING (A.2) LSTM data processing methods

A.3 LSTM model definition

```

1
2 import os
3 import math
4 import numpy as np
5 import datetime as dt
6 from numpy import newaxis
7 from core.utils import Timer
8 from keras.layers import Dense, Activation, Dropout, LSTM
9 from keras.models import Sequential, load_model
10 from keras.callbacks import EarlyStopping, ModelCheckpoint
11
12 class Model():
13     """A class for an building and inferencing an lstm model"""
14
15     def __init__(self):
16         self.model = Sequential()
17
18     def load_model(self, filepath):
19         print('[Model] Loading model from file %s' % filepath)
20         self.model = load_model(filepath)
21
22     def build_model(self, configs):
23         timer = Timer()
24         timer.start()
25
26         for layer in configs['model']['layers']:
27             neurons = layer['neurons'] if 'neurons' in layer else None
28             dropout_rate = layer['rate'] if 'rate' in layer else None
29             activation = layer['activation'] if 'activation' in layer else None
30             return_seq = layer['return_seq'] if 'return_seq' in layer else None
31             input_timesteps = layer['input_timesteps'] if 'input_timesteps' in
layer else None
32             input_dim = layer['input_dim'] if 'input_dim' in layer else None
33
34             if layer['type'] == 'dense':
35                 self.model.add(Dense(neurons, activation=activation))
36             if layer['type'] == 'lstm':
37                 self.model.add(LSTM(neurons, input_shape=(input_timesteps,
input_dim), return_sequences=return_seq))
38             if layer['type'] == 'dropout':
39                 self.model.add(Dropout(dropout_rate))
40
41         self.model.compile(loss=configs['model']['loss'], optimizer=configs['
model']['optimizer'])
42
43         print('[Model] Model Compiled')
44         timer.stop()
45
46     def train(self, x, y, epochs, batch_size, save_dir):
47         timer = Timer()
48         timer.start()
49         print('[Model] Training Started')
50         print('[Model] %s epochs, %s batch size' % (epochs, batch_size))
51
52         save_fname = os.path.join(save_dir, '%s-e%s.h5' % (dt.datetime.now().
strftime('%d%m%Y-%HPM%S'), str(epochs)))
53         callbacks = [
54             EarlyStopping(monitor='val_loss', patience=2),
55             ModelCheckpoint(filepath=save_fname, monitor='val_loss',
save_best_only=True)

```



```

56     ]
57     self.model.fit(
58         x,
59         y,
60         epochs=epochs,
61         batch_size=batch_size,
62         callbacks=callbacks
63     )
64     self.model.save(save_fname)
65
66     print('[Model] Training Completed. Model saved as %s' % save_fname)
67     timer.stop()
68
69     def train_generator(self, data_gen, epochs, batch_size, steps_per_epoch,
70         save_dir):
71         timer = Timer()
72         timer.start()
73         print('[Model] Training Started')
74         print('[Model] %s epochs, %s batch size, %s batches per epoch' % (
75             epochs, batch_size, steps_per_epoch))
76
77         save_fname = os.path.join(save_dir, '%s-e%s.h5' % (dt.datetime.now().
78             strftime('%d/%m/%Y-%H%M%S'), str(epochs)))
79         callbacks = [
80             ModelCheckpoint(filepath=save_fname, monitor='loss', save_best_only=
81                 True)
82         ]
83         self.model.fit_generator(
84             data_gen,
85             steps_per_epoch=steps_per_epoch,
86             epochs=epochs,
87             callbacks=callbacks,
88             workers=1
89         )
90
91         print('[Model] Training Completed. Model saved as %s' % save_fname)
92         timer.stop()
93
94     def predict_point_by_point(self, data):
95         #Predict each timestep given the last sequence of true data
96         print('[Model] Predicting Point-by-Point...')
97         predicted = self.model.predict(data)
98         predicted = np.reshape(predicted, (predicted.size,))
99         return predicted
100
101     def predict_sequences_multiple(self, data, window_size, prediction_len):
102         #Predict sequence of 50 steps before shifting prediction run forward
103         #by 50 steps
104         print('[Model] Predicting Sequences Multiple...')
105         prediction_seqs = []
106         for i in range(int(len(data)/prediction_len)):
107             curr_frame = data[i*prediction_len]
108             predicted = []
109             for j in range(prediction_len):
110                 predicted.append(self.model.predict(curr_frame[newaxis, :, :])[0,0])
111                 curr_frame = curr_frame[1:]
112                 curr_frame = np.insert(curr_frame, [window_size-2], predicted[-1],
113                     axis=0)
114             prediction_seqs.append(predicted)
115         return prediction_seqs
116
117     def predict_sequence_full(self, data, window_size):

```

```
112 #Shift the window by 1 new prediction each time, re-run predictions on
    new window
113 print('[Model] Predicting Sequences Full...')
114 curr_frame = data[0]
115 predicted = []
116 for i in range(len(data)):
117     predicted.append(self.model.predict(curr_frame[newaxis, :, :])[0,0])
118     curr_frame = curr_frame[1:]
119     curr_frame = np.insert(curr_frame, [window_size-2], predicted[-1],
    axis=0)
120 return predicted
```

LISTING (A.3) LSTM model class object

Appendix B

Natural Language Processing

B.1 Stop Words

```
1 ‘‘
2 =
3 >
4 |
5 -
6 —
7 ;
8 :
9 !
10 ?
11 .
12 ‘’
13 ‘
14 ”
15 ""
16 (
17 )
18 [
19 ]
20 @
21 $
22 *
23 **
24 \&
25 \%
26 +
27 -
28 +1
29 10-k
30 10-q
31 1/2
32 +1-646-223-8780
33 1-brazil
34 1-china
35 1-u.s.
36 2016
37 2017
38 2018
39 21st
40 2-u.s.
41 3d
42 3rd
43 4th
44 +91
45 a
46 'a
```

47 'a-
48 'a+
49 a.
50 A
51 'aa
52 'aaa
53 ab
54 abb
55 about
56 abu
57 africa
58 african
59 after
60 ago
61 al
62 alberta
63 alibaba
64 all
65 allergan
66 also
67 amazon
68 amazon.com
69 america
70 american
71 amsterdam
72 an
73 and
74 angeles
75 announce
76 another
77 anthem
78 any
79 apollo
80 apple
81 apply
82 april
83 are
84 area
85 around
86 as
87 a/s
88 asia
89 ask
90 astrazeneca
91 at
92 aug
93 aug.
94 august
95 australia
96 australian
97 automaker
98 aviv
99 b
100 'b
101 'b+
102 b.
103 baidu
104 bancorp
105 bangalore.newsroom
106 bank
107 barclays
108 bb
109 'bb

110 'bb-
111 'bb+
112 'bbb
113 'bbb-
114 'bbb+
115 bce
116 be
117 because
118 become
119 before
120 begin
121 beijing
122 bell
123 billion
124 billionaire
125 billiton
126 blackstone
127 bln
128 bloomberg
129 boeing
130 book
131 boston
132 both
133 brazil
134 brazilian
135 brief
136 brief-aegon
137 brief-american
138 brief-taiwan
139 britain
140 british
141 brussels
142 bt
143 business
144 by
145 c
146 c.
147 ca
148 calgary
149 california
150 can
151 canada
152 canadian
153 car
154 carolina
155 case
156 cent
157 century
158 -ceo
159 certain
160 cf
161 chase
162 chevron
163 chicago
164 china
165 cigna
166 cisco
167 cite
168 citi
169 citigroup
170 cme
171 cnbc
172 co

173 co .
174 coca-cola
175 colombia
176 comcast
177 company
178 corp
179 corp .
180 corporate
181 corporation
182 customer
183 d .
184 dan
185 data
186 date
187 day
188 day-
189 de
190 dec
191 dec .
192 december
193 dell
194 delta
195 detroit
196 deutsche
197 disney
198 do
199 doj
200 dollar
201 don
202 donald
203 dow
204 dr .
205 dubai
206 during
207 dutch
208 e
209 e .
210 east
211 eight
212 eikon
213 el
214 eli
215 e*trade
216 eu
217 euro
218 europe
219 europe
220 european
221 event
222 exchange
223 exxon
224 f
225 ' f1
226 ' f2
227 f-35
228 facebook
229 fargo
230 fast-food
231 f/cast
232 feb
233 february
234 fell
235 ffo

236 field
237 file
238 final
239 firm
240 first
241 first-quarter
242 fitch
243 five
244 fla .
245 florida
246 for
247 ford
248 four
249 fourth
250 fourth-quarter
251 fox
252 france
253 francisco
254 frankfurt
255 french
256 friday
257 from
258 full-year
259 further
260 fy
261 fy2016
262 g
263 gaap
264 ge
265 george
266 germany
267 get
268 give
269 glaxosmithkline
270 gm
271 gmt
272 goldman
273 google
274 gopro
275 gp
276 green
277 group
278 h
279 h1
280 ha
281 halt
282 has
283 have
284 hbo
285 he
286 his
287 holiday
288 home
289 honda
290 houston
291 how
292 hsbc
293 http
294 humana
295 i
296 i/b/e/s
297 ibm
298 icahn

299 icici
300 idr
301 if
302 ifr
303 ii
304 immuno-oncology
305 in
306 inbev
307 inc
308 inc.
309 index
310 india
311 indian
312 indonesia
313 international
314 into
315 iphone
316 is
317 israeli
318 it
319 italia
320 item
321 its
322 iv
323 j
324 j.
325 jan.
326 january
327 japan
328 japanese
329 j.c.
330 jeffrey
331 job
332 john
333 johnson
334 journal
335 jp
336 j.p.
337 jpmorgan
338 jr.
339 july
340 june
341 just
342 jv
343 k
344 kellogg
345 kfc
346 korea
347 l
348 l.
349 l-3
350 le
351 let
352 letter
353 lg
354 likely
355 link
356 llc
357 lng
358 local
359 local/gmt
360 lockheed
361 london

362 look
363 lp
364 l.p.
365 l.p.
366 ltd
367 ltd.
368 lynch
369 m
370 m.
371 main
372 make
373 march
374 market
375 may
376 mcdonald
377 merrill
378 mexico
379 michael
380 microsoft
381 million
382 mln
383 mobil
384 monday
385 monsanto
386 moody
387 morgan
388 mt
389 must
390 m/v
391 mylan
392 name
393 nasdaq
394 near
395 netflix
396 next
397 nike
398 nikkei
399 nine
400 no.2
401 nokia
402 non-timely
403 nordisk
404 norm
405 north
406 northern
407 norwegian
408 note
409 nov
410 nov.
411 novo
412 now
413 nv
414 nyse
415 o
416 obama
417 oct
418 october
419 of
420 oi
421 on
422 one
423 or
424 other

425 our
426 over
427 p
428 p.
429 pa
430 parent
431 part
432 partner
433 past
434 paulo
435 pc
436 pct
437 pdvsa
438 per
439 percent
440 period
441 petrobras
442 pfizer
443 phase
444 place
445 plc
446 point
447 poland
448 previously
449 q1
450 q2
451 q3
452 q4
453 qtrly
454 quarter
455 quarterly
456 r
457 r.
458 ralph
459 recent
460 recently
461 region
462 relate
463 report
464 reports
465 result
466 reuters
467 richard
468 rico
469 rig
470 rio
471 russian
472 s
473 's
474 s.
475 sa
476 s.a.
477 san
478 sanction
479 sao
480 saturday
481 say
482 says
483 se
484 season
485 seattle
486 sec
487 -sec

488 second
489 second-quarter
490 sector
491 sedan
492 sept
493 sept.
494 september
495 series
496 service
497 set
498 shanghai
499 she
500 shr
501 singapore
502 six
503 sixth
504 size
505 sk
506 so
507 so-called
508 some
509 son
510 sony
511 soon
512 -sources
513 south
514 southern
515 sport
516 squibb
517 st
518 st.
519 stanley
520 start-up
521 state
522 still
523 stocks-tsx
524 stocks-wall
525 struggle
526 such
527 suisse
528 sunday
529 sweeten
530 swiss
531 sydney
532 t
533 t.
534 taiwan
535 take
536 tap
537 td
538 tel
539 telecom
540 telefonica
541 tell
542 ten
543 tesla
544 teva
545 thai
546 than
547 that
548 the
549 their
550 them

551 there
552 they
553 third
554 third-quarter
555 this
556 thomson
557 thomsonreuters.com
558 those
559 through
560 thursday
561 t-mobile
562 to
563 tokyo
564 too
565 toronto
566 total
567 toyota
568 transcanada
569 tuesday
570 tv
571 twitter
572 two
573 type
574 u
575 ubs
576 uk
577 unit
578 us
579 u.s.
580 u.s.
581 usa
582 u.s.-based
583 user
584 u.s.-listed
585 utility
586 v
587 verizon
588 version
589 versus
590 viacom
591 visa
592 volkswagen
593 volume
594 vs.
595 w
596 w.
597 wa
598 wall
599 wal-mart
600 walt
601 warner
602 was
603 washington
604 we
605 web
606 website
607 wednesday
608 week
609 wells
610 were
611 west
612 what
613 when

```
614 where
615 whether
616 which
617 who
618 whose
619 will
620 williams
621 with
622 within
623 wo
624 world
625 would
626 wsj
627 x
628 xl
629 year
630 year-ago
631 yet
632 york
633 you
634 y/y
635 zuckerberg
```

LISTING (B.1) Stop Words

B.2 Punctuation List

```
1 PUNC_LIST = [
2     "." ,
3     "!" ,
4     "?" ,
5     "," ,
6     ";" ,
7     ":" ,
8     "-" ,
9     "/" ,
10    "\"" ,
11    "!!" ,
12    "!!!" ,
13    "???" ,
14    "???" ,
15    "?!?" ,
16    "!!?" ,
17    "?!?!" ,
18    "!!?!" ,
19 ]
```

LISTING (B.2) VADER punctuation list

B.3 Negate List

```
1 NEGATE = {
2     "aint" ,
3     "arent" ,
4     "cannot" ,
5     "cant" ,
6     "couldnt" ,
7     "darent" ,
8     "didnt" ,
9     "doesnt" ,
10    "ain't" ,
11    "aren't" ,
```

```

12 "can ' t " ,
13 "couldn ' t " ,
14 "daren ' t " ,
15 "didn ' t " ,
16 "doesn ' t " ,
17 "dont " ,
18 "hadnt " ,
19 "hasnt " ,
20 "havent " ,
21 "isnt " ,
22 "mightnt " ,
23 "mustnt " ,
24 "neither " ,
25 "don ' t " ,
26 "hadn ' t " ,
27 "hasn ' t " ,
28 "haven ' t " ,
29 "isn ' t " ,
30 "mightn ' t " ,
31 "mustn ' t " ,
32 "neednt " ,
33 "needn ' t " ,
34 "never " ,
35 "none " ,
36 "nope " ,
37 "nor " ,
38 "not " ,
39 "nothing " ,
40 "nowhere " ,
41 "oughtnt " ,
42 "shant " ,
43 "shouldnt " ,
44 "uhuh " ,
45 "wasnt " ,
46 "werent " ,
47 "oughtn ' t " ,
48 "shan ' t " ,
49 "shouldn ' t " ,
50 "uh-uh " ,
51 "wasn ' t " ,
52 "weren ' t " ,
53 "without " ,
54 "wont " ,
55 "wouldnt " ,
56 "won ' t " ,
57 "wouldn ' t " ,
58 "rarely " ,
59 "seldom " ,
60 "despite " ,
61 }

```

LISTING (B.3) VADER negate list

B.4 Booster Dictionary

```

1 BOOSTER_DICT = {
2     "absolutely": B_INCR,
3     "amazingly": B_INCR,
4     "awfully": B_INCR,
5     "completely": B_INCR,
6     "considerably": B_INCR,
7     "decidedly": B_INCR,

```

```
8 "deeply": B_INCR,
9 "effing": B_INCR,
10 "enormously": B_INCR,
11 "entirely": B_INCR,
12 "especially": B_INCR,
13 "exceptionally": B_INCR,
14 "extremely": B_INCR,
15 "fabulously": B_INCR,
16 "flipping": B_INCR,
17 "flippin": B_INCR,
18 "fricking": B_INCR,
19 "frickin": B_INCR,
20 "frigging": B_INCR,
21 "friggin": B_INCR,
22 "fully": B_INCR,
23 "fucking": B_INCR,
24 "greatly": B_INCR,
25 "hella": B_INCR,
26 "highly": B_INCR,
27 "hugely": B_INCR,
28 "incredibly": B_INCR,
29 "intensely": B_INCR,
30 "majorly": B_INCR,
31 "more": B_INCR,
32 "most": B_INCR,
33 "particularly": B_INCR,
34 "purely": B_INCR,
35 "quite": B_INCR,
36 "really": B_INCR,
37 "remarkably": B_INCR,
38 "so": B_INCR,
39 "substantially": B_INCR,
40 "thoroughly": B_INCR,
41 "totally": B_INCR,
42 "tremendously": B_INCR,
43 "uber": B_INCR,
44 "unbelievably": B_INCR,
45 "unusually": B_INCR,
46 "utterly": B_INCR,
47 "very": B_INCR,
48 "almost": B_DECR,
49 "barely": B_DECR,
50 "hardly": B_DECR,
51 "just enough": B_DECR,
52 "kind of": B_DECR,
53 "kinda": B_DECR,
54 "kindof": B_DECR,
55 "kind-of": B_DECR,
56 "less": B_DECR,
57 "little": B_DECR,
58 "marginally": B_DECR,
59 "occasionally": B_DECR,
60 "partly": B_DECR,
61 "scarcely": B_DECR,
62 "slightly": B_DECR,
63 "somewhat": B_DECR,
64 "sort of": B_DECR,
65 "sorta": B_DECR,
66 "sortof": B_DECR,
67 "sort-of": B_DECR,
68 }
```

LISTING (B.4) VADER booster dictionary

B.5 Special case idioms

```
1 SPECIAL_CASE_IDIOMS = {  
2     "the shit": 3,  
3     "the bomb": 3,  
4     "bad ass": 1.5,  
5     "yeah right": -2,  
6     "cut the mustard": 2,  
7     "kiss of death": -1.5,  
8     "hand to mouth": -2,  
9 }
```

LISTING (B.5) VADER special case idioms

Bibliography

- Aggarwal, Charu C (2014). *Data classification: algorithms and applications*. CRC press.
- Arlot, Sylvain, Alain Celisse, et al. (2010). "A survey of cross-validation procedures for model selection". In: *Statistics surveys* 4, pp. 40–79.
- Bengio, Yoshua, Patrice Simard, Paolo Frasconi, et al. (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166.
- Berners-Lee, Tim, James Hendler, Ora Lassila, et al. (2001). "The semantic web". In: *Scientific american* 284.5, pp. 28–37.
- Boeing, Geoff and Paul Waddell (2017). "New insights into rental housing markets across the united states: web scraping and analyzing craigslist rental listings". In: *Journal of Planning Education and Research* 37.4, pp. 457–476.
- Chaulagain, Ram et al. (2017). "Cloud Based Web Scraping for Big Data Applications". In: pp. 138–143. DOI: [10.1109/SmartCloud.2017.28](https://doi.org/10.1109/SmartCloud.2017.28).
- Cox, M. and D. Ellsworth (1997). "Application-controlled demand paging for out-of-core visualization". In: pp. 235–244. DOI: [10.1109/VISUAL.1997.663888](https://doi.org/10.1109/VISUAL.1997.663888).
- Crane, DJ (1997). "How the Web is changing the business of business information." In: *Electronic Library* 15.4, pp. 311–316. ISSN: 0264-0473. URL: <https://www.emeraldinsight.com/doi/abs/10.1108/eb045575>.
- Cybenko, George (1989). "Approximations by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2, pp. 183–192.
- Denning, PJ (1990). "Saving all the bits". In: *American Scientist* 78.5, pp. 402–405. ISSN: 00030996. URL: <https://ntrs.nasa.gov/search.jsp?R=19910023503>.
- Ferrara, Emilio et al. (2014). "Web data extraction, applications and techniques: A survey". In: *Knowledge-based systems* 70, pp. 301–323.
- Hackeling, Gavin (2017). *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd.
- Hanin, Boris (2017). "Universal function approximation by deep neural nets with bounded width and relu activations". In: *arXiv preprint arXiv:1708.02691*.
- Hawkins, Douglas M (2004). "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1, pp. 1–12.
- Hutto, Clayton J and Eric Gilbert (2014). "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Eighth international AAAI conference on weblogs and social media*.
- Johnston, Alan B and Daniel C Burnett (2012). *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC.

- Jothimani, Dhanya, Ravi Shankar, and Surendra S Yadav (2018). "A big data analytical framework for portfolio optimization". In: *arXiv preprint arXiv:1811.07188*.
- Karlik, Bekir and A Vehbi Olgac (2011). "Performance analysis of various activation functions in generalized MLP architectures of neural networks". In: *International Journal of Artificial Intelligence and Expert Systems* 1.4, pp. 111–122.
- Kirkpatrick II, Charles D and Julie A Dahlquist (2010). *Technical analysis: the complete resource for financial market technicians*. FT press.
- Kleene, Stephen C (1956). "Automata studies". In:
- Krotov, Vlad and Matthew Tennyson (2018a). "Tutorial: Web Scraping in the R Language". In:
- (2018b). "Tutorial: Web Scraping in the R Language". In:
- Laney, Doug (2001). "3D data management: Controlling data volume, velocity and variety". In: *META group research note* 6.70, p. 1.
- Lantz, Brett (2015). *Machine learning with R*. Packt Publishing Ltd.
- Lawson, Richard (2015). *Web scraping with Python*. Packt Publishing Ltd.
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, p. 436.
- Lesk, Michael (1997). "Digital Libraries: A Unifying or Distributing Force?." In: URL: <https://eric.ed.gov/?id=ED414929>.
- McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.
- Michael, Katina and Keith W Miller (2013). "Big data: New opportunities and new challenges [guest editors' introduction]". In: *Computer* 46.6, pp. 22–24.
- Saldaña, Zoe Wilkinson (2018). "Sentiment Analysis for Exploratory Data Analysis". In: URL: <https://programminghistorian.org/en/lessons/sentiment-analysis>.
- Shah, Vatsal H (2007). "Machine learning techniques for stock prediction". In: *Foundations of Machine Learning | Spring* 1.1, pp. 6–12.
- Trifunovic, Nemanja et al. (2015). "Paradigm Shift in Big Data SuperComputing: DataFlow vs. ControlFlow". In: *Journal of Big Data* 2.1, p. 4. ISSN: 2196-1115. DOI: [10.1186/s40537-014-0010-z](https://doi.org/10.1186/s40537-014-0010-z). URL: <https://doi.org/10.1186/s40537-014-0010-z>.
- Zhu, Xiaojin and Andrew B Goldberg (2009). "Introduction to semi-supervised learning". In: *Synthesis lectures on artificial intelligence and machine learning* 3.1, pp. 1–130.

Listings

3.1	A simple socket	17
3.2	Webpage retrieval	17
3.3	S&P500 Tickers	21
3.4	S&P500 Stock data	21
3.5	News data scraping utilities	23
3.6	Reuters news crawler	24
3.7	Joined close prices	27
4.1	Sentiment Intensity Analyzer	48
5.1	LSTM settings	54
A.1	Reuters crawler class object	66
A.2	LSTM data processing methods	69
A.3	LSTM model class object	71
B.1	Stop Words	74
B.2	VADER punctuation list	84
B.3	VADER negate list	84
B.4	VADER booster dictionary	85
B.5	VADER special case idioms	87