Pontifícia Universidade Católica
do Rio de Janeiro

João Paulo Forny de Melo

Predicting Trends in the Stock Market

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Ruy Luiz Milidiú

Rio de Janeiro
February 2018

Pontifícia Universidade Católica
do Rio de Janeiro

João Paulo Forny de Melo

Predicting Trends in the Stock Market

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Prof. Ruy Luiz Milidiú
Advisor
Departamento de Informática – PUC-Rio

Prof. Hélio Cortês de Vieira Lopes
Departamento de Informática – PUC-Rio

Prof. Marco Antonio Casanova
Departamento de Informática – PUC-Rio

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico – PUC-Rio

Rio de Janeiro, February 27th, 2018

João Paulo Forny de Melo

Graduaded in computer science by the Federal Fluminense University (UFF). His research is focused in Machine Learning and Natural Language Processing.

Bibliographic data

Melo, João Paulo Forny de

Predicting Trends in the Stock Market / João Paulo Forny de Melo; advisor: Ruy Luiz Milidiú. – 2018.
54 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2018.
Inclui bibliografia

1. Informática – Teses. 2. Aprendizado de Máquina;. 3. Mercado de Ações;. 4. Google Trends;. 5. Predição de Séries Temporais.. I. Milidiú, Ruy Luiz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Thanks to my advisor Prof. Ruy Luiz Milidíu for his friendship, endless support and encouragement for this work.

Thanks to PUC-Rio and CAPES, for the financial support which allowed this work to be done.

Thanks to the friends of LEARN, for their support and friendship. To all colleagues, faculty and staff of the Department of PUC-Rio, for the fellowship, learning and support.

To my mother, who is no longer with us, but always supported and guided me to pursue a research oriented career.

Abstract

Melo, João Paulo Forny de; Milidiú, Ruy Luiz (Advisor). **Predicting Trends in the Stock Market**. Rio de Janeiro, 2018. 54p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Investors are always looking for an edge. However, traditional economic theories tell us that trying to predict short-term stock price movements is wasted effort, since it approximate a random walk, i.e., a stochastic or random process. Besides, these theories state that the market is efficient enough to always incorporate and reflect all relevant information, making it impossible to "beat the market". In recent years, with the growth of the web and data availability in conjunction with advances in Machine Learning, a number of works are using Natural Language Processing to predict share price variations based on financial news and social networks data. Therefore, strong evidences are surfacing that the market can, in some level, be predicted. This work describes the development of an application based on Machine Learning to predict trends in the stock market, i.e., positive, negative or neutral price variations with minute granularity. We evaluate our system using B3 (Brasil Bolsa Balcão), formerly BM&FBOVESPA, stock quotes data, and a dataset with the most relevant topics of Google Search and its related articles, provided by the Google Trends platform and collected, minute by minute, from 08/15/2016 to 07/10/2017. The experiments show that this data provides useful information to the task at hand, in which we achieve 69.24% accuracy predicting trends for the PETR4 stock, creating some leverage to make profits possible with intraday trading.

Keywords

Machine Learning; Stock Market; Google Trends; Time Series Forecasting.

Resumo

Melo, João Paulo Forny de; Milidiú, Ruy Luiz. **Predizendo Tendências na Bolsa de Valores**. Rio de Janeiro, 2018. 54p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Investidores estão sempre à procura de uma vantagem. Porém, tradicionais teorias financeiras nos dizem que tentar predizer tendências na bolsa de valores é um esforço em vão, uma vez que seguem um passeio aleatório, i.e., um processo estocástico ou randômico. Além disso, afirma-se que o mercado é eficiente de maneira que sempre incorpora e reflete toda informação relevante, o que torna impossível 'bater o mercado". Recentemente, com o crescimento da web e aumento da disponibilidade de dados em conjunto com a evolução dos algoritmos de Aprendizado de Máquina, diversos trabalhos tem aplicado técnicas de Processamento de Linguagem Natural em notícias financeiras e dados de redes sociais para prever variações do preço de ações. Consequentemente, estão surgindo fortes evidências que o mercado pode, em algum grau, ser previsto. Este trabalho descreve o desenvolvimento de uma aplicação baseada em Aprendizado de Máquina para realizar a predição de tendências no mercado de ações, i.e., variações negativas, positivas ou neutras de preços com granularidade de minuto. Avaliamos o sistema usando dados de cotação de ações da B3 (Brasil Bolsa Balcão), antiga BM&FBOVESPA, e um dataset de tópicos mais relevantes buscados no Google Search e seus artigos relacionados, que são disponibilizados pela plataforma Google Trends e coletados, minuto a minuto, de 15/08/2016 até 10/07/2017. Os experimentos mostram que esses dados provêem informação relevante para a tarefa em questão, onde conseguimos uma acurácia de 69.24% para a predição de tendências do ativo PETR4, criando alguma vantagem para tornar possível lucrar com negociações intraday.

Palavras-chave

Aprendizado de Máquina; Mercado de Ações; Google Trends; Predição de Séries Temporais.

Table of contents

List of figures

List of tables

List of Abreviations

AI – Artificial Intelligence
ML – Machine Learning
EMH – Efficient Market Hypothesis
B3 – Brasil Bolsa Balcão
NLP – Natural Language Processing
NN – Neural Networks
ANN – Artificial Neural Networks
MLP – Multi Layer Perceptron
BOW – Bag of Words
CBOW – Continuous Bag of Words
TF-IDF – Term Frequency-Inverse Document Frequency
SVM – Support Vector Machines
RNN – Recurrent Neural Networks
LSTM – Long Short-Term Memory Networks

1
Introduction

1.1
Overview

Investor are always looking for an edge. However, traditional economic theories tells us that trying to predict short-term stock price movements is a wasted effort, since it approximate a random walk, i.e., a stochastic or random process. This concept was introduced in 1965, by the American economist Eugene Fama, also known as "The Father of Finance", in his Ph.D. thesis, entitled "The Behavior of Stock Market Prices" (1). Subsequently, that work was rewritten into a less technical article (2). Later, he proposed the concept of the Efficient Market Hypothesis (EMH) (4). According to the EMH, the market is efficient enough to always incorporate and reflect all relevant information. Thus, since news are unpredictable, share prices will follow a random walk pattern and should not be predictable with an accuracy higher than 50%, making it impossible to "beat the market".

These concepts have been extensively tested throughout the following years and studies were mostly in favor of Fama's statements (6, 7). Recently, as data availability has reached considerable high volume, velocity and variety (Big Data), with advances in Machine Learning, strong evidences are surfacing that the market can, in some level, be predicted (8, 10). In a review, even Fama (5) states that the EMH must be false. Furthermore, a number of works are using Natural Language Processing to predict share price variations based on financial news and social networks data (9, 11).

It is a fact that news may be unpredictable, but that very early indicators can be extracted from online social media to predict changes in various economic and commercial indicators. For instance, Google search queries have been shown to provide early indicators of disease infection rates and consumer spending (14). Influenza epidemics was also predicted with search query data (15). This may also be conceivably the case for the stock market.

Although news most certainly influence stock market prices, public sentiment may play an equally important role. However, to analyze how public mood influences the stock markets, we need reliable, scalable and early

assessments of this data at a time-scale and resolution appropriate for practical stock market prediction. To address that problem, we created a dataset with the most relevant topics of Google Search, called Trending Stories, provided by the Google Trends platform, from 08/15/2016 to 07/10/2017. We only selected stories from the Brazil region and the Business category.

This dissertation starts from the assumption that the trending topics in Google Search are highly correlated with stock price changes and provide some insight into future trends, since these topics reflect important events with regard to public's attention. We analyze the influence of this data against B3 (Brasil Bolsa Balcão), formerly BM&FBOVESPA, stock exchange share prices.

This dissertation describes the development of an application based on Machine Learning to predict trends in the stock market, i.e., positive, negative or neutral price variations with minute granularity. The experiments show that this data provide useful information to the task at hand, creating some leverage to make profits possible with intraday trading.

1.2 Related Work

The increasing volumes of data reflecting various aspects of our everyday activities has opened up new options for researchers. Stock market prediction are certainly a prime target for such investigations. Several recent works are based on the assumption that these new data sources resulting from human interaction offer valuable perspectives on the behaviour of the market.

In 2011, at PUC-Rio, Paula Sonnenfeld (17) constructed a dataset with news articles related to Petrobras in order to analyze the sentiment with regard to the stock market, i.e., if the article was favorable or not for the company, reaching a 87.14% accuracy on the task at hand. Later in 2014, Heraldo Borges (18) followed Paula's work with an augmented dataset, but developed a classifier to predict high, low or neutral variations of Petrobras stock PETR4, achieving a 68.57% accuracy.

In 2014 and 2015, Ding et al. (12, 13) extracted structured events from Reuters and Bloomberg news articles titles in order to predict daily movements - increase or decrease - of the S&P 500 index.

Although news may provide information about what affects stock price movements, they mostly have a crucial lag from the actual moment of the reported event. However, this delay is considered very small in social networks like Twitter, which have gained much attention in recent years.

Bollen et al. (9) investigated whether public mood as measured from large-scale collection of tweets is correlated or even predictive of Dow Jones

Industrial Average (DJIA) values. Twitter messages are read by an algorithm which distinguishes between calm, alert, sure, vital, kind and happy messages. The main finding of this work is that some of the mood intensities can help explain the variation in the market index.

The ability of Google Trends to reveal information about financial markets has also been of recent interest. To uncover the relationship between the volume of search queries for a specific term and the overall direction of trader decisions, Preis et al. (16) analyzed the Google search query volumes from 2004 to 2011 for a set of 98 mostly finance-related search terms, such as *debt*, *investment*, *bonds*, and so on. By looking at how stock prices changed over that same time, they attempted to find out search patterns that showed "early warning signs" of market moves. They also tested trading strategies that would act on these signs. The authors also show that the word *debt* is the one with the best overall results, and one trading plan based on changes in searches for this term would have yielded a return of 326 percent over the period analyzed. By comparison, a "buy and hold" investment in the DJIA yielded 16 percent return. This work was extended by Damien Challet in 2013 (51).

In this year, Hongping et al. (52) studied the application of Neural Networks to predict daily directions of opening prices of the S&P 500 and DJIA indices. They used Google Trends search volume index of the keywords "S&P 500" and "DJIA" alongside the opening price, the highest price, the lowest price, the closing price, and the trading volume of the current trading day. Their results show that Google Trends can help in predicting the direction of the stock market index with a slightly accuracy increase.

All precedent works found in the literature use the search volume index provided by Google Trends, which is related to specific terms. This dissertation use Trending Stories, which represent a collection of topics, aggregated by Google itself.

1.3 Contribution and Outline

In this dissertation we develop models for predicting trends in the stock market. In particular, we develop classification models that process Google Trends data in addition to the numerical price time series and predict whether the opening price of the next minute will be higher, lower or equal to the current minute.

In Chapter 2, we describe how the world is changing due to the current possibilities for storing and analyzing data, as well as exploring the impact of social media. Then we present details of Machine Learning with regard to

the overall process of building an intelligent application. This is followed by a historical description of Neural Networks up to the one of the most recent breakthrough in the field, Deep Learning.

Chapter 3 contains an explanation of the main problem tackled in this work, which is to predict trends in the stock market. The main goal is to analyze the influence of Google Trends trending stories of the business category and the Brazil region with regard to the task at hand. We selected stocks from B3, formely BM&FBOVESPA, which is Brazil's most important stock exchange.

Chapter 4 presents the methodology used to build the classification models. The total size of the constructed dataset is approximately 86.000 minutes or 180 days. We use the skip-gram word2vec algorithm to train a text representation model to incorporate semantic information of trending stories.

Chapter 5 contains the experimental evaluation of the two implemented models. As baseline, we trained a Linear Support Vector Classification model and later analyzed its performance against a Long-Short Term Memory network, which has been extensively used for time series forecasting and overall sequence problems.

Finally, Chapter 6 presents the achievements of this dissertation and identifies remaining challenges and potential avenues of evaluation in future work.

2
Background

In recent years, Artificial Intelligence (AI) has again become notoriously important in Computer Science. It was formally named by John McCarthy in 1956 after writing a proposal for a project about machines that reason intelligently (19). Currently, the research area has many subfields, varying from the general-purpose like logical reasoning to specific areas such as playing chess or disease diagnosis. Scientists from other fields frequently move into AI, finding tools to computationally express intellectual tasks, and automate and improve their work (20). Besides, Artificial Intelligence adopters can apply their methods to a huge variety of areas related to knowledge extraction and representation. Hence, it is truly a universal field.

Much of the revival in interest in AI is due to Machine Learning, the part of AI responsible for developing computational theories focused on data pattern extraction in order to make predictions or decisions. Software developed for this technology has the characteristic of making decisions based on previously accumulated knowledge. Hence, within some context, one can take a history of actions of objects of interest, learn from this record, and then model these activities to improve our understanding of this context going forward in the future. The advent of Machine Learning is a product of the collapsing costs of information processing and the growing masses of digital data that can now be gathered and processed online.

2.1
Big Data

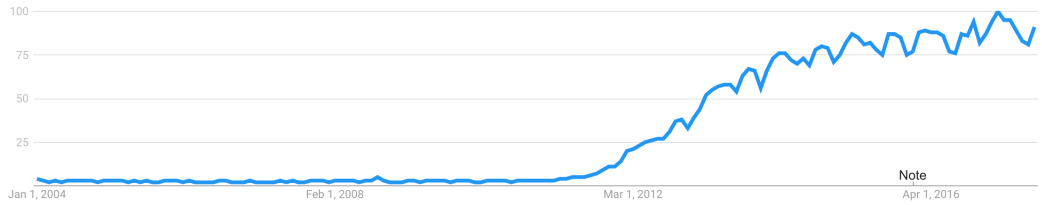
In Man's unquenchable search for wisdom and understanding, the extraction of information from the analysis of observable data has been the foundation of experimental science. This data is all around us and the larger the dataset, the more accurate any analysis is likely to be. This will help decision makers to be better informed and confident to make the appropriate decisions such as improving operational efficiencies or reducing costs and risk.

The world has been gradually generating more and more data in smaller and smaller time frames. In 2011, it was estimated that 90 per cent of data in circulation had been generated in the preceding two years (32).

Big Data is a popular term used to describe this exponential growth and availability of data. The term Big Data itself is not new, but is now receiving more attention, given that we have enough cheap storage capabilities, sensors and technology to capture almost every conceivable type of data, as illustrated in Figure 2.1, where the y-axis represents the relative number of searches for the term Big Data in relation to the total number of searches on Google over the last 10 years. However, the biggest challenge to be tackled in this field is how to explore this almost unimaginable volume of data and extract value.

Data Science is the integration of methods from statistics, computer science, and other fields for gaining insights from data. In practice, data science encompasses an iterative process of data harvesting, cleaning, analysis and visualization, and implementation. Analyzing and extracting actionable insight has became the complex field within Data Science known as Data Analytics, which is based on Machine Learning and Natural Language Processing (NLP) techniques. For instance, IBM and New York Genome Center (NYGC) have a partnership to assist doctors in the treatment of cancer, analyzing DNA mutations in order to develop personalized treatments for each patient. IBM also has a partnership with Repsol to use Big Data to optimize reservoir production and enhance decision making when acquiring new oil fields. Walmart and other big retail stores are using data captured by sensors, social networks and other sources to better understand customer behavior and therefore meet their clients needs more accurately.

Figure 2.1: Search Interest for Big Data



Source: Google Trends, January 1, 2004 – September 29, 2017, Indexed Search Query Volume, Worldwide

While unstructured data refers to information that either does not have a pre-defined data model or is not organized in a predefined manner, structured data is that kind that can be easily organized. Unstructured data is primarily human-generated, which reflects into the heavy role that social media plays in the generation of unstructured data.

This new era of Big Data is therefore now also regimented by behavior of social networks and their users, thus requiring far more complex analysis of

those status updates, comments, photos, videos, and so on. Nowadays social networks have become one of the most significant tools for rapid diffusion of information. Huge amounts of data, generated daily, need to be analyzed in real time or near real time to define influence, reach and relevancy to a user, to understand the context of the data or simply identify the user's sentiment about a specific topic.

2.2 Machine Learning

Machine Learning uses data in a concise and productive way, considering only relevant information and automatically ignoring outliers. It has the ability to *train* machines to extract patterns from previous data, acting in an almost human-like fashion in certain contexts, to be able to build self-driving cars or to analyze tweets, for example. While this approach might not be completely correct all of the time, the solutions are often considered good enough to solve everyday problems. In practice, this approach requires data and, more than often, a lot of data.

It is important to make clear the basic difference between Machine Learning and Data Mining. The two concepts are complementary, but also overlap. The main focus of Machine Learning is to make predictions based on previously known data that is used to build a model or in other words "train" the algorithm. The main focus of Data Mining is the discovery of properties of the data that were not previously known. For this reason, Machine Learning may use Data Mining to improve the accuracy of the model training and assessment.

It is natural to conclude that, with the advent of Big Data, the possibilities of creating such models are significantly enhanced and thus the correctness of their predictions too. Clearly, with more available data, it is probable that the predictions will be better. With the addition of real time information sources and real time decision making continuously creating even more data, there is a rapidly growing amount of data in our daily lives that influences our choices and behavior. But how can this data and information be easily understood and translated to improve the efficiency or quality of our business processes and our lives?

2.2.1 General Concepts of Machine Learning

There are so many concepts, theories and algorithms surrounding this field that is very difficult to mention or summarize them all (21). This section

aims at describing, in a very abstract way, the sequence of steps that needs to be addressed in order to create a Machine Learning-based application (22).

1. **Data Selection:** identifying the data that be used to achieve the desired goal is critical. This process involves cleaning, sorting and analyzing the suitability of the data to be used. Without the appropriate data, it becomes difficult to make accurate predictions.

2. **Feature Selection:** A feature is a measurable property of a dataset, for instance, to a computer, an image is just a bunch of pixel values. From a Machine Learning point of view, often the pixel values are the features, but higher level features can be extracted, such as average pixel intensity or detecting the edges and counting how many edges exist in the top quadrant of the image. Not only is selecting the best features of the data important, equally crucial can be the ones less sensitive to noise or simply those easier to extract. This is also the step where the data sets for model training - the examples the application will learn from - and model testing - the examples that will be used to verify the accuracy of the model - are defined.

3. **Model Selection:** One should start by developing simple models and increase their complexity, only if necessary. The model is part of a past reality where the occurrence of what took place is known. This allows one to perform the following phases of training and testing to see if the proposed algorithm is able to predict the outcome with the required quyality level.

4. **Learning:** The training phase is of great importance where the main objective is to configure and tune the algorithm parameters in order to minimize the errors.

5. **Evaluation:** This is the test phase where a data set, independent from the one used for learning, is used. If the algorithm has a very large error, it is inevitable that the model will need to be reviewed, modified and the learning phase performed again.

6. **Application:** This step is where the model is applied to new data and analyzed to see if the results are consistent or not with reality. If the overall results are not acceptable, the algorithm will have to be better trained, performing step 4 again.

7. **Production:** At this point, the model has already been successfully validated and the time has come to put everything into production. Of

course, even in production, the model should be continuously tuned and evolved in concert with changes in the processes being modelled.

2.2.1.1 Neural Networks

Artificial Neural Networks are mathematical models that simulate the operation of Biological Neural Networks. They are based on research into the structure and functions of the human brain, and try to emulate intelligence with regard to information processing. Some studies of neurophysiology consider that the computational power and efficiency of the human brain is associated with its large number of neurons, interconnected by a complex network of synapses. It is estimated that the number of neurons of the human brain is of the order of billions. However, the processing speed of these components is relatively slow as compared with traditional computers. This lack of speed is overcome by the huge quantity of neurons operating in parallel. Such characteristics allow the human brain to execute some functions, such as sound and image recognition, in a way that conventional computers aren't able to compute with similar performance (23).

2.2.1.2 Biological Neural Networks

A biological neuron is composed of the fiollowing main items: dendrites, which are designed to receive the stimuli transmitted by other neurons; the neuron body, also called soma, which is responsible for collecting and combining information coming from other neurons; and finally, the axon, which consists of a tubular fiber, is responsible for transmitting the stimuli to other cells via synapses that connect neurons.

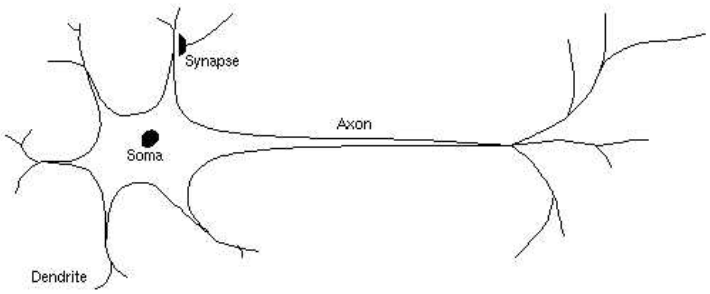


Figure 2.2: Biological Neuron Structure

In other words, a neuron communicates with many other neurons through synapses with information being received by the respective dendrites and

passed on by the axon. Dendrites receive nervous impulses from other neurons that are carried on to the cell body or soma. The information received by each synapse is aggregated and if this sum exceeds a certain threshold, the axon transmits the stimulus to other neurons. The synaptic strength of the neural connections, which reflect the level of excitation or inhibition between adjacent neurons, enables the human brain to store knowledge and consequently learn. Across synapses, neurons collectively provide functionality forming neural networks (24).

2.2.1.3 Artificial Neural Networks

The first work on Artificial Neural Networks (ANNs) was carried out by McCulloch and Pitts in 1943 (25) where they studied biological neuron behavior in order to model it mathematically. The result of this research was of great importance for future works regarding neuron implementations. First, the neuron activity is all-or-nothing, i.e., the neuron will be in an active state if its output surpass a given threshold, otherwise, it will stay in an idle state. An active state means that the neuron is transmitting its output to others neurons in the network. Second, the activity of any inhibitory synapse prevents the excitement of the neuron at that moment. The second statement was important in the building of the formal neuron based on the concept of weights, that is, each neuron entry will have an associated value; if positive, it will tend to excite the cell; and if it is negative, it will tend to inhibit. Neurons produce only binary outcomes and connections transmit only zeros and ones. Their networks are composed of weighted connections of excitatory and inhibitory type. Each unit is characterized by certain threshold and to determine the state of a neuron a synaptic integration is made, i.e., the weighted sum of input states with their respective synaptic weights is calculated. If the sum is greater than or equal to a certain threshold, initially fixed, the neuron becomes active and the output is a pulse, otherwise it remains inactive. With enough of these simple units (neurons) and synaptic connections properly adjusted and operating synchronously, McCulloch and Pitts showed that a network thus constituted could achieve, in principle, the representation of any computable function.

The next significant development in ANNs came in 1949, when Hebb (27) presented a explicit formulation of a philosophical learning rule for synaptic modification. In that work, Hebb hypothesized that the connectivity of the brain is continuously modified as the organism keeps learning in order to perform new tasks and that neuron groups are created by such modifications.

He demonstrated that the learning ability in biological neural networks comes from changes in synaptic efficiency between cells, which is increased due to repetitive activations of a neuron by another neuron. Below is the famous Hebb postulate:

"Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

The first Neural Network model implementation was the perceptron, by Frank Rosenblatt in 1958 (28). The perceptron is based on the McCulloch and Pitts neuron with an input and output layer. For each entry, there exists a related weight wherein the output value is the sum of the products of each entry with their respective weight. Rosenblatt describes a topology for Neural Networks and proposes an algorithm to train the network and execute certain types of functions. The perceptron behaves as a pattern classifier, dividing the entry space into distinct regions that represent each existing class.

Despite the impact the perceptron had on the Artificial Intelligence community, this model was heavily criticized by Minsky and Papert (26). In their book, the authors cite an example of the perceptron with a single layer that cannot simulate, for instance, the behavior of a simple XOR function (exclusive-or), since it can only solve linearly separable[1] problems. However, after the publication of this work, the period known as the dark years of the neural networks started, as discouraging new research, a lack of funding and deeper studies turned interest away from this field.

The deficiency of the perceptron with non-linear patterns was eliminated by Rumelhart, Hinton and Willians (30). The solution was the generalization of the Delta Rule, known as the Backpropagation algorithm. The perceptron had proved its validity, enabling the implementation of the third layer required to learn the XOR. Using a network of neurons as those used in the perceptron, the backpropagation performs a back-propagation of the output error for prior layers. The error is the result of the comparison between the desired output (pre-determined) and the actual output of the network. With this back-propagation, in conjuction with one threshold function of fractional values

[1]A dataset is said to be linearly separable if there exists a hyperplane in the input space that separates data from different classes. In geometry, a hyperplane can be a vector space, affine transformation or n-1 dimensional subspace. For example, for a tridimensional dataset, the hyperplane is a normal 2D plane, for a bidimensional, a line and for a unidimensional, a point.

(leaving out the all-or-nothing), representation of non-linear functions was made possible, allowing a multi layer perceptron (MLP) Neural Network be trained to learn the XOR function.

2.2.1.4 Neural Network Characteristics

The operation model of a processing unit (Neuron) proposed by McCulloch and Pitts can be summarized as follows:

- Signals are provided as input;
- Each signal is multiplied by a number, or weight, which indicates or reflects their influence on the output;
- The weighted sum of the signals produces a level of activity;
- If this activity level exceeds a certain threshold, the unit outputs 1, otherwise, 0.

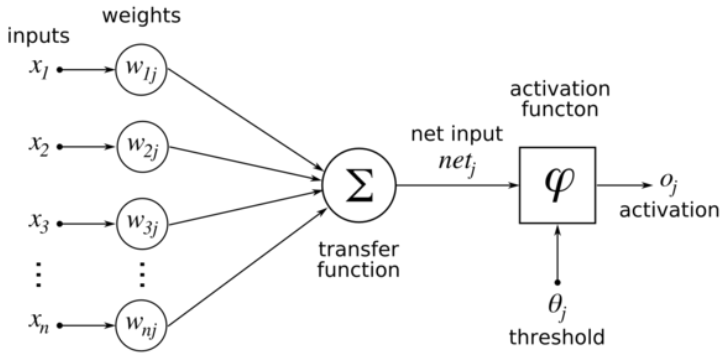


Figure 2.3: Artificial Neuron Structure

A perceptron takes several binary inputs, x_1, x_2, ..., and produces a single binary output. The basic mathematical model can be described as follows:

$$output = \begin{cases} 0 & if \sum_j w_j x_j \leq threshold \\ 1 & if \sum_j w_j x_j > threshold \end{cases}$$

To simplify, some notational changes can be made. One can write $\sum_j w_j x_j$ as a dot product, $w \cdot x \equiv \sum_j w_j x_j$, where w and x are vectors whose components are the weights and inputs, respectively, and move the threshold to the other side of the inequality, replacing it by what's known as the perceptron's bias, $b \equiv -threshold$. Hence, the perceptron rule can be redefined as:

$$output = \begin{cases} 0 & if w \cdot x + b \leq 0 \\ 1 & if w \cdot x + b > 0 \end{cases}$$

Perceptrons can be used to represent basic elementary logical functions such as AND, OR, and NAND. For example, if we create a perceptron with two inputs in the input layer, each with weight -2, and an overall bias of 3. In a simple way, Figure 2.4 describes this perceptron.

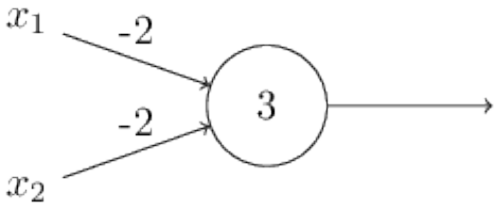


Figure 2.4: Simple perceptron with weigths and bias

Analyzing the possible outcomes, the input 00 produces output 1, since $(-2)\times 0+(-2)\times 0+3 = 3$ is positive. Similarly, inputs 01 and 10 produce output 1. But the input 11 produces output 0, since $(-2) \times 0 + (-2) \times 0 + 3 = -1$ is negative. This lets us conclude that the perceptron above implements a NAND gate. This example shows that we can use perceptrons to compute simple logical functions. In fact, we can use networks of perceptrons to compute any logical function since the NAND gate is universal for computation, that is, we can implement any computation out of NAND gates.

The computational universality of perceptrons mean that that networks of perceptrons can be as powerful and flexible as any other computing device. But it is also disappointing, because it makes it seem as though perceptrons are merely a new type of NAND gate. However, it turns out that we can use learning algorithms which can automatically tune the weights and biases of a network of artificial neurons without the direct intervention of a programmer. These learning algorithms enable us to use artificial neurons in a way which is radically different to conventional logic gates. Instead of explicitly laying out a circuit of NAND and other gates, our neural networks can simply learn to solve problems, sometimes problems where it would be extremely difficult to directly design a conventional circuit.

Most Neural Network models have some form of training step, where the weights of connections are adjusted in accordance with the provided patterns. In other words, they learn through examples. Neural architectures are typically organized into layers that usually are classified as shown below:

- **Input layer:** Where inputs are presented to the network;
- **Intermediate or Hidden Layers:** Where most of the processing is done through weighted connections; Can be considered as a feature extractor;

– **Output layer:** Where the output is produced.

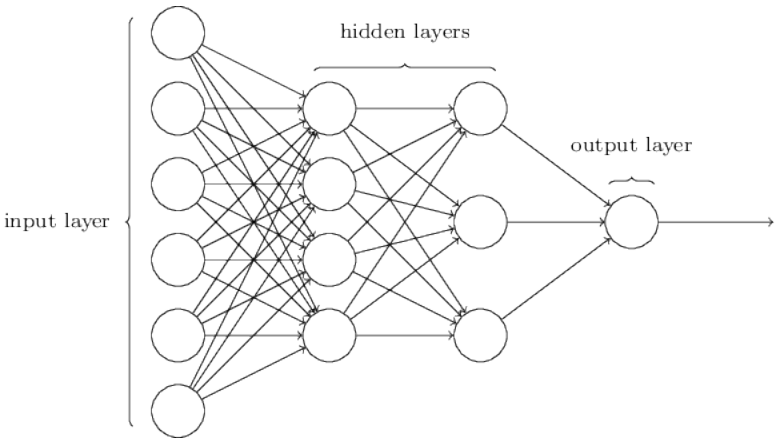


Figure 2.5: A Multi-layered Neural Network

In Figure 2.5, a Neural Network with four layers and two hidden layers is presented. The term “hidden” refers to the layers between the input and output ones. Designing input and output layers is often simple. For instance, suppose we want a Neural Network to recognize the digit “9". A good way to design the input layer is to encode image pixels intensities into the input neurons. The output layer will contain just a single neuron, with output value indicating if the image is a “9" or not. However, designing the hidden layers has been a major interest of Neural Network research. While there aren’t a few simple rules of thumb, one of the most crucial trade offs is related to the number of hidden layers against the time required to train the network.

2.2.1.5 Deep Learning

A subarea of Machine Learning, known as deep learning, accounts for much of the renewed excitement surrounding AI. The depth of an architecture is related to the number of levels of non-linear operations in the learned function. In a Neural Network, this corresponds to the number of layers. Functions that can be represented by an architecture of depth k may require an exponential number of computational elements to be represented by an architecture of level $k-1$. This means that some real world functions cannot be represented by shallow architectures.

Deep Learning methods focus on learning features of higher levels through the composition of lower level features and is inspired by the human brain. For decades, researchers tried, without success, to train Neural Networks with multiple deep layers. But in 2006, Hinton discovered that the results of a deep Neural Network could be improved when pre-trained with an

unsupervised learning algorithm layer by layer, starting with the first one (31). This work started the area known today as Deep Learning.

2.3 Recurrent Neural Networks

The motivation behind Recurrent Neural Networks (RNNs) is to use sequential information. In traditional neural networks we assume that all inputs and outputs are independent of each other. Moreover, vanilla NNs accept a fixed-sized vector as input and produce a fixed-sized vector as output, e.g, class probabilities, and perform this mapping using a fixed amount of steps, e.g., the number of layers of the NN.

RNNs process sequences in the input, the output, or both, performing the same task for every element of the input, with the output being depended on the previous computations. A few examples are shown below in Figure 2.6. Each rectangle is a vector and arrows represent functions. Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state. Notice that in every case there are no pre-specified constraints on the sequence length because the recurrent transformation (green) is fixed and can be applied as many times as we like.

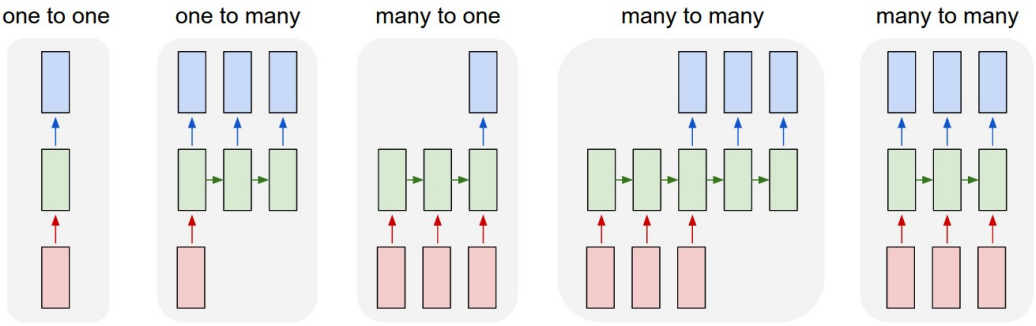


Figure 2.6: Sequence Problems Diagrams (36). From left to right: (1) Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). (2) Sequence output (e.g. image captioning takes an image and outputs a sentence of words). (3) Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). (4) Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). (5) Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).

The vanilla RNN has a simple form. Basically, it processes a sequence of vectors $x_1, ..., x_2$ applying the recurrence $h_t = f_\theta(h_{t-1}, x_t)$ and the same parameters θ for every time step. The state vector h_t can be considered as a context of the input until that timestep, which is updated by the recurrence

formula shown below. These equations omit the additional bias vector for brevity.

$$h_t = tanh\left(W\begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}\right)$$

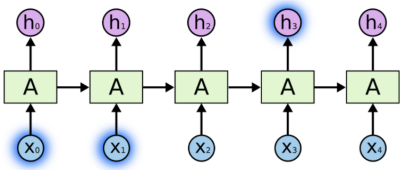


Figure 2.7: Short-term dependency.

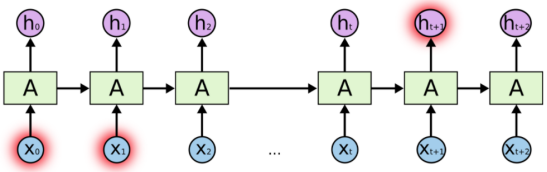


Figure 2.8: Long-term dependency.

In the above diagram, a chunk of neural network, A, looks at some input x_t and outputs a value h_t, also passing it to the next step. Also, consider a language model that predicts the next work based on the previous ones. For example, in Figure 2.7, we are trying to predict the last word in the phrase "the clouds are in the *sky*". The network would need only to look at three words backwards in the sequence. RNNs can learn these short-term dependencies, i.e., problems where the gap between the relevant information and the place that it's need is small.

However, for the sentence "I grew up in France... I speak fluent French.", as illustrated in Figure 2.8, this gap can become very large. Long-term dependencies should also be handled by RNNs, but in practice, they are not able to learn them. Unfortunately, the vanilla RNN formulation leads to undesirable dynamics during backpropagation (37, 38). In summary, the gradients tend to either vanish or explode over long time periods, since RNNs maintain a vector of activations for each timestep, which makes them extremely deep.

To address that problem, the Long Short-term Memory (LSTM) network was proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber (34), which was refined and improved in 2000 by Felix Gers (35). Its recurrence formula has a form that allows the inputs x_t and h_{t-1} interact in a more computationally complex manner that includes multiplicative interactions, and the LSTM recurrence uses additive interactions over time steps that more effectively propagate gradients backwards in time (34):

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$h_t = o \odot tanh(c_t)$$

$$c_t = f \odot c_{t-1} + i \odot g$$

In addition to the state vector h_t, LSTMs also have a memory vector c_t. They have a chain like structure just like RNNs, but instead of a single neural network layer, there are four. Considering that the hidden state has H units, the three vectors $i, f, o \in \mathbb{R}^H$ are thought of as binary gates that control whether each memory cell is updated, whether it is reset to zero, and whether its local state is revealed in the hidden vector, respectively. The activations of these gates are based on the sigmoid function and hence allowed to range smoothly between zero and one to keep the model differentiable. The vector $g \in \mathbb{R}^H$ ranges between -1 and 1 and is used to additively modify the memory contents.

In summary, LSTM cells internally decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies.

2.4 Text Representation

Throughout history, many models have been explored to tackle the problem of text representation, which is a task required for many Natural Language Processing (NLP) problems such as language modeling and classification. The most traditional method to represent documents is known as bag-of-words (BOW), commonly applied in conjunction with term frequency-inverse document frequency (TF-IDF) weighting, which intends to reflect how important a word is to a document in a collection or corpus, storing information about the presence or absence of tokens in a fixed length vector. These models perform well in several multi-class classification problems, but they lack awareness of word semantics and suffer from sparsity and high dimensionality, since they consider words as atomic symbols, i.e., words are represented as indexes in a dictionary and hence there is no notion of similarity between them. Moreover, word order in a document are completely ignored.

There are an estimated 13 million tokens for the English language which obviously are not completely unrelated. To address that, Bengio et

al. (39) introduced the first distributional neural word embedding model, which relies on the Distributional Hypothesis (33), i.e., words that appear in the same contexts share semantic meaning. This approach was a breakthrough in language modeling and attracted much attention back in 2003, but was limited due to its training complexity. Later, Mikolov et al. (41) proposed two novel architectures to compute continuous representations of words that is computationally efficient, allowing to learn word vectors from huge data sets, i.e., billions of words and million of words in the vocabulary. In summary, words are encoded in low dimensional space, commonly between $\mathbb{R}^{100}$ and $\mathbb{R}^{300}$, based on the notion of local context, i.e., preceding and succeeding words, where semantically similar words are embedded nearby each other.

Despite the success of Mikolov's model, known as *word2vec*, dealing with variable length pieces of text was still a problem, since many classification algorithms require the input to be represented as a fixed length vector. In 2014, Le and Mikolov (40) proposed the Paragraph Vector model, also known as *paragraph2vec*, an unsupervised method for learning distributed representations for pieces of texts, such as sentences, paragraphs, and documents, overcoming the weaknesses of BOW models.

2.4.1 Atomic Word Representation

The atomic word representation, used in the BOW model, considers words as one-hot vectors. Basically, given a vocabulary V, every word is a $\mathbb{R}^{|V|\times 1}$ vector where its value is 1 at the index of the word and 0 at the rest. We represent each word as if they were completely unrelated, and thus there is no notion of similarity between them. This served as inspiration to the development of models with reduced dimensions and encoding of relationships between words.

2.4.2 Vector Representations of Words

Here we describe the two model architectures for computing continuous vector representations of words from very large data sets proposed by Mikolov et al. (41). The explanations below are based on the ones given by Rong (42).

2.4.2.1 Continuous Bag of Words

The objective of this model is to predict the target word given context words, i.e., nearby words in a local window.

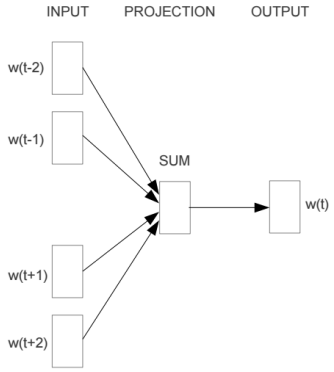


Figure 2.9: Continuous Bag of Words model architecture. The CBOW architecture predicts the current word based on the context.

To illustrate how the vectors are computed, we will explain the simplest version of CBOW, i.e., one-word context, which means that the model will predicted a target word given a context, as shown in Figure 2.10. The network layers are fully connected. The vocabulary size is V , and the hidden layer size is N. The input is a one-hot encoded vector.

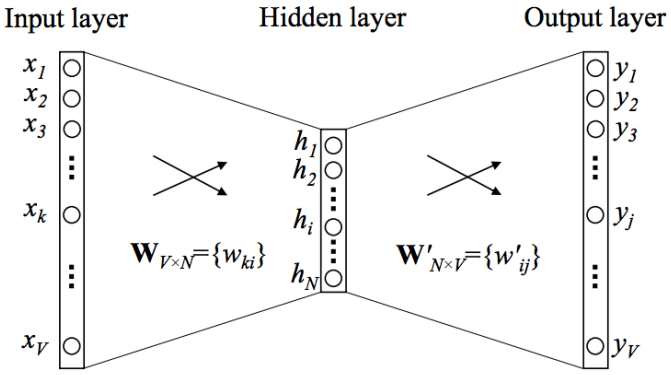


Figure 2.10: Simple CBOW model with context size of 1 word.

The weights are represented as a matrix W and W', both with size $N \times V$. Each row of W is the vector representation of a word v_w in the vocabulary. Formally, row i of W is v_w^T. Given a context (single word), we can assume $x_k = 1$ and $x_{k'} = 0$ for $k' \neq k$. Thus, we have

$$h = W^T x = W^T_{(k,.)} := v^T_{w_I}, \tag{2-1}$$

which is esseantially copying the k-th row of W to h. v_{w_I} is the vector representation of the input word w_I. In summary, the hidden layer is directly

passing its weighted sum of inputs to the next layer.

From the hidden layer to the output layer, we can compute a score u_j for each word in V,

$$u_j = v'^T_{w_j} h, \tag{2-2}$$

where v'_{w_j} is the j-th column of W' matrix. Then, we wish to obtain the posterior distribution of words, which is a multinomial distribution can be done with softmax, a log-linear classification model,

$$p(w_j|w_I) = y_j = \frac{exp(u_j)}{\sum_{j'=1}^{|V|} exp(u_{j'})}, \tag{2-3}$$

where y_j is the output of the j-th unit in the output layer. Substituting (1) and (2) into (3), we obtain

$$p(w_j|w_I) = \frac{exp(v'^T_{w_j} v_{w_I})}{\sum_{j'=1}^{|V|} exp(v'^T_{w_{j'}} v_{w_I})}, \tag{2-4}$$

Note that v_w and v'_w are two different representations of the word w, which came from W and W' respectively. Empirically, the representation v_w has showed better performance and is the one adopted in the literature. It is also important to highlight that in practice, hierarchical softmax is preferred to softmax for fast training.

By feeding this neural network architecture with a large dataset, it can be trained using stochastic gradient descent where the gradient is obtained via backpropagation. After the training converges, words with similar meaning are mapped to a similar position in the vector space. For example, “powerful" and “strong" are close to each other, whereas “powerful" and “Paris" are more distant. The difference between word vectors also carry meaning. For example, the word vectors can be used to answer analogy questions using simple vector algebra: “$king$” − “man” + “$woman$” = “$Queen$” (43). It is also possible to learn a linear matrix to translate words and phrases between languages (44).

Analogously, this simple approach can be extended for bigger contexts. Figure 2.11 shows the CBOW model with a multi-word context setting. When computing the hidden layer output, instead of directly copying the input vector of the input context word, the CBOW model takes the average of the vectors of the input context words, and use the product of the W weight matrix and the average vector as the output. C are the number of words in the context.

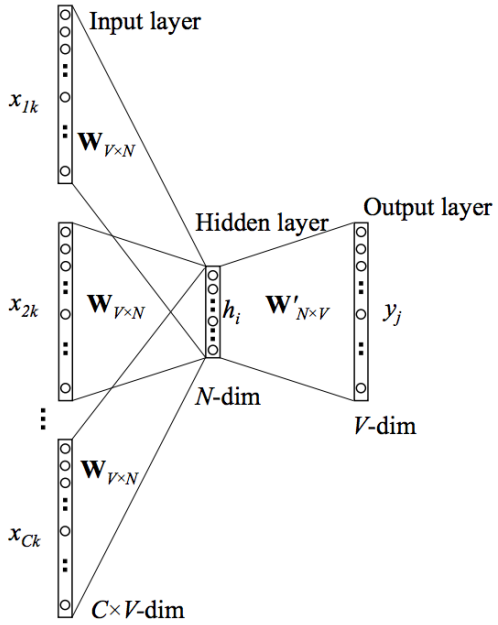


Figure 2.11: CBOW model for context size of C words.

2.4.2.2 Skip-gram

In this model, the objective is to predict context words given a target word.

We will not dive into the mathematics of this model, since we consider that a good intuition was given in the CBOW section and understanding it will surely made things easier regarding other neural word embedding models. Hence, only a superficial analysis of the main idea behind the model is presented. Figure 2.13 shows the skip-gram model in details. It is the opposite of the CBOW model. The target word is now at the input layer, and the context words are on the output layer.

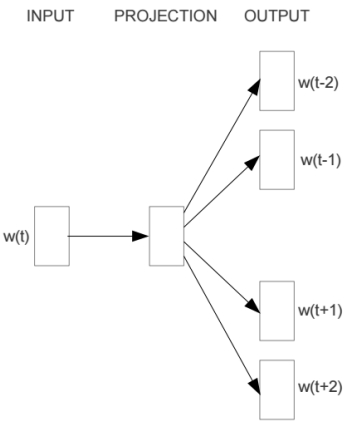


Figure 2.12: Skip-gram model architecture. The training objective is to learn word vector representations that are good at predicting the nearby words.

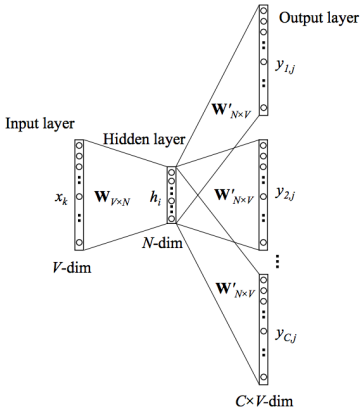


Figure 2.13: Skip-gram model architecture.

3
Trend Prediction

This chapter describes and analyzes the problem of predicting trends in the Brazilian stock market. Generally, a *trend* can be defined as "a general direction in which something is developing or changing". In other words, we tackle the challenge of predicting price variations with minute granularity, i.e., for the given minute, the classifier should be able to output if the price in the next minute will have a negative, positive or neutral variation.

3.1
B3

The B3 (Brasil Bolsa Balcão S.A.), formely BM&FBOVESPA, is Brazil's most important stock exchange, located in São Paulo. B3 was born from the merge of BM&FBOVESPA with CETIP at March 30, 2017. The company is now the fifth larger stock exchange in the world, with a market capitalization of 13 billion dollars. The benchmark indicator of BM&FBOVESPA is the *Índice Bovespa*, which comprises the most representative companies in the market, both by market capitalization and traded volume.

B3 provides daily, tick by tick market data of its stocks via ftp[1], which includes both orders and negotiations data. Basically, in a stock exchange, a negotiation is completed when a buy order is matched against a sell order. An order is an instruction to trade at the specified price or better and will stand as an offer to trade until someone is willing to trade at its limit price, until it expires or is cancelled. Below we present the layout of the negotiation data.

[1] ftp://ftp.bmf.com.br/MarketData

Column	Description
Session Date	Negotiation date
Instrument Symbol	Instrument identifier, .e.g, PETR4
Trade Number	Trade identifier
Trade Price	Trade price
Traded Quantity	Traded quantity
Trade Time	Trade time (format HH:MM:SS.NNNNNN)
Trade Indicator	Trade indicador: 1 - Trade / 2 - Trade cancelled
Buy Order Date	Buy order date
Sequential Buy Order Number	Sequential buy order number
Secondary Order ID - Buy Order	Secondary Order ID - Buy Order.
Aggressor Buy Order Indicator	0 - Neutral (Order was not executed) / 1 - Aggressor / 2 - Passive
Sell Order Date	Sell order date
Sequential Sell Order Number	Sequential sell order number
Secondary Order ID - Sell Order	Secondary Order ID - Sell Order.
Aggressor Sell Order Indicator	0 - Neutral (Order was not executed) / 1 - Aggressor / 2 - Passive
Cross Trade Indicator	Define if the cross trade was intentional: 1 - Intentional / 0 - Not Intentional
Buy Member	Entering Firm (Buy Side) - Available from March/2014
Sell Member	Entering Firm (Sell Side) - Available from March/2014

Table 3.1: Tick data layout.

Among this data, the columns of interest for this work is the session date and the trade time, the instrument symbol, which is an identifier of a company's stock, the trade price, traded quantity and trade indicator.

3.2 Google Trends

Google Trends is a public web platform based on Google Search. Its main functionality, called "Interest over time", shows relative significance of one or more search-terms. Basically, it is a time series chart where numbers represent search interest relative to the highest point for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. Likewise a score of 0 means the term was less than 1% as popular as the peak. For illustration purposes, we present below the interest over time for the topics "Machine learning", "Deep learning" and "Natural language processing", all categories worldwide for the past 12 months.

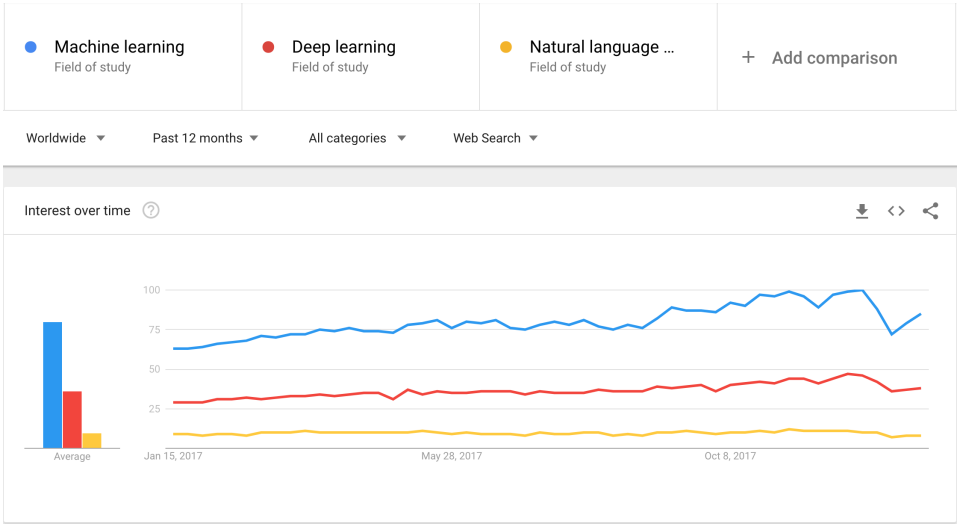


Figure 3.1: Interest Over Time for the Topics Machine learning, Deep learning and Natural language processing.

On the Google Trends homepage you can explore Trending Stories in real time by category and location. A trending story is a collection of Knowledge Graph topics, search interest, trending YouTube videos or Google News articles.

The Google Knowledge Graph is a system that Google launched in May 2012 that aims to understand facts about real-world things and places and how these entities are all connected. It is used both behind-the-scenes to help Google improve its search relevancy and also to present Knowledge Graph boxes, at times, within its search results to provide direct answers. It was estimated that by May 2016, knowledge boxes were appearing for roughly one-third of the estimated 100 billion monthly searches the company processed.

Trending Stories rely on technology from the Knowledge Graph across Google Search, Google News, and YouTube to detect when topics are trending on these three platforms. The algorithm for trending stories groups topics together and ranks stories based on the relative spike in volume and the absolute volume of searches. Figure 3.2 shows a screenshot of the top 8 trending stories at 7PM of 01/16/2018. It can be noticed that the top story is the one with the most recent spike in search volume, shown in the small blue time series charts on the right.

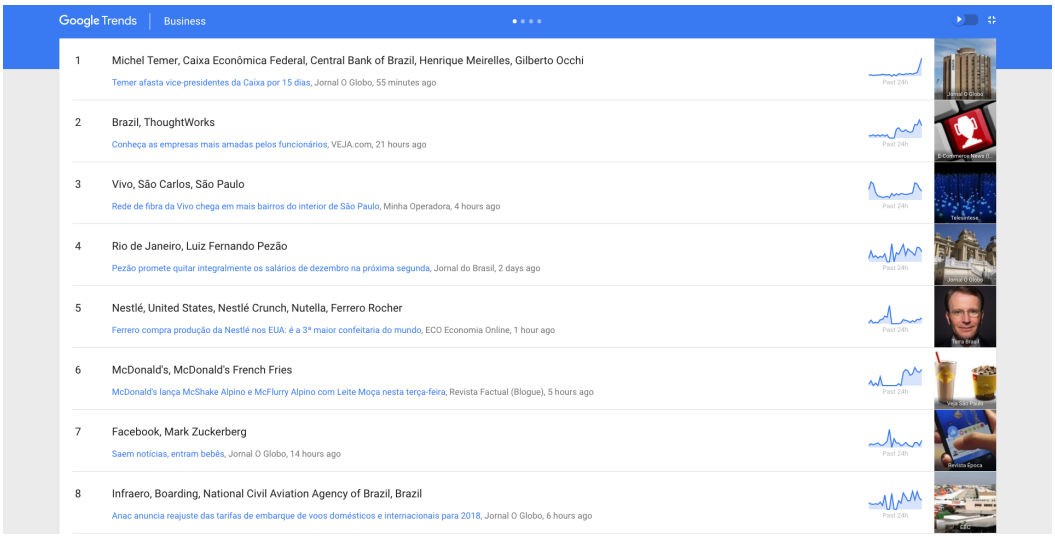


Figure 3.2: Trending Stories of the business category and Brazil location.

A story contains data like the most relevant articles, interest over time, interest by region, trending queries and related topics. The former two are shown below in Figure 3.3.

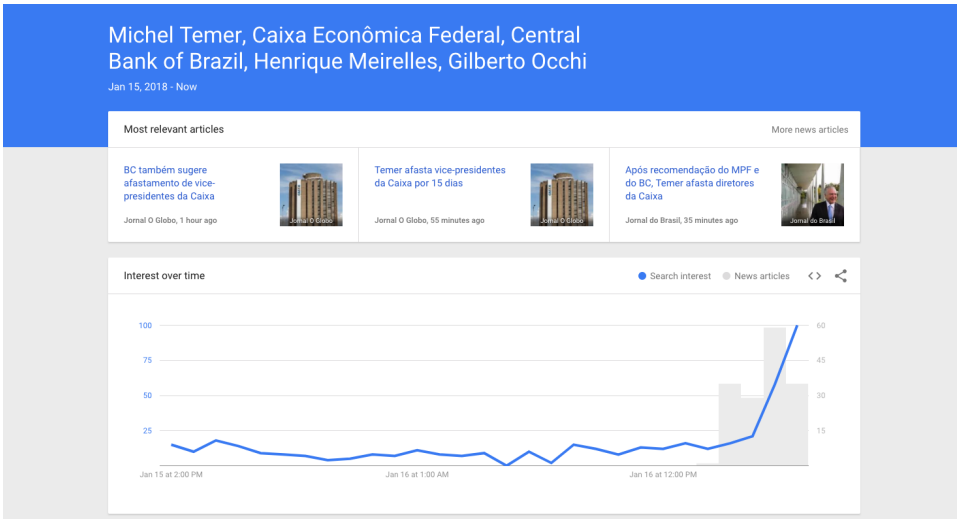


Figure 3.3: Trending Story's most relevant articles and interest over time.

In the "Interest Over Time" chart, the bar chart represents the number of Google News articles written per hour and corresponds to the grey axis on the right. The line graph represents the Google Search interest over time and corresponds to the blue axis on the left. The most relevant articles comes from Google News' full coverage of stories, using its ranking algorithm to select the top articles for that trending story.

For this work, we collected the list trending stories of the business category and the Brazil region, more specifically the trending entities and its most related articles titles.

3.3 Time Series Forecasting

Time series forecasting is a classic problem in many domains, with wide-ranging and high-impact applications. The central problem of forecasting is that of predicting the value $Y(T+1)$ given past observations $Y(1), ..., Y(T)$. However, there are many ways to reframe a forecast problem, which can both simplify the prediction problem and potentially expose relevant data that was not modeled. In addition, reframing may provide a suite of highly related problems, tackled with diverse perspectives. This process can result in models that capture different information from the input but maintain a satisfactory performance. Hence, the models can be combined into an ensemble to make more robust forecasts.

In this work, we will consider the time series forecast problem as a classification problem, since we consider that a real-valued output is not necessary to identify a trend. In summary, we tackle the problem to predict, using trending stories past observations additional data, whether the value $Y(T+1)$ will be higher, lower or equal to $Y(T)$.

4
Methodology

In this chapter we present the methodology used to build the stock trend predictor, i.e., the constructed dataset, text representation models and classification algorithms.

4.1
Dataset

The dataset used in this work basically consists of stock prices and trending stories, collected from B3 ftp and Google Trends website respectively.

The trends dataset contains the list of trending stories and its most relevant articles for every minute between 5AM and 9PM, collected from 08/15/2016 to 07/10/2017. The data was crawled directly from Google Trends through one of its API endpoint[1] with the same parameters used by the website frontend.

The stock quotes dataset provided by B3 contains data for every negotiation as described in Table 3.1. This data was then aggregated by minute to match trending stories granularity, i.e., we computed the average, close, high and low prices, number of ticks, traded quantity and volume. We selected stocks based on market capitalization, which is calculated from the share price (as recorded on selected day) multiplied by the number of outstanding share. The top 5 companies in April 2017 are listed below in Table 4.1[2].

Company	Instrument Symbol	MarketValue
Ambev	ABEV3	R$ 276.6 billions
Itaú	ITSA4	R$ 231.4 billions
Petrobras	PETR3, PETR4	R$ 195.7 billions
Bradesco	BBDC3, BBDC4	R$ 176.5 billions
Vale	VALE3, VALE4	R$ 144.9 billions

Table 4.1: Top 5 B3 Companies in April 2017

We selected the top 3 companies to conduct further experiments. The total size of the dataset is approximately 86.000 minutes or 180 days.

[1]https://www.google.com/trends/api/stories/latest

[2]http://www.gazetadopovo.com.br/economia/nova-economia/conheca-as-20-maiores-empresas-de-capital-aberto-do-brasil-axgiib32w169d97e0uce56a7n

4.2 Text Representation

To incorporate semantic information of trending stories entities and its related articles, we put the good old bag of words aside and choose a word2vec approach, since it can learn good quality continuous vector representations of words on a large amount of data. Hence, we used all Wikipedia and Wikinews articles written in portuguese to generate word embeddings. Wikipedia is a global, free and multilingual internet encyclopedia, on the other hand, Wikinews is a free-content news source wiki. Dumps from both of these sites are provided by the Wikimedia Foundation[3].

To train the word2vec model, we used the implementations provided by gensim (46). Gensim is an open-source vector space modeling and topic modeling toolkit written in Python. The chosen model was the skip-gram, which is the task of predicting the context for a given word, i.e., the words before and after the target within a window. The choices of hyperparameters were based on the original paper, since the authors showed its good performance in the semantic analogy task. We used negative sampling to train the model and a context window of size 5.

In Figure 4.1, we show the embedding of the word *petrobras* and its 100 nearest neighbours. We can note that relevant information is represented by the model. For instance, the words *petrolifera* and *petroleo*, which describes the nature of the current entity, are very close to the target word. Also, it is related to the words *fraudar* and *propina*, probably because of the recent corruption scandal involving the company.

[3]https://dumps.wikimedia.org/ptwiki/

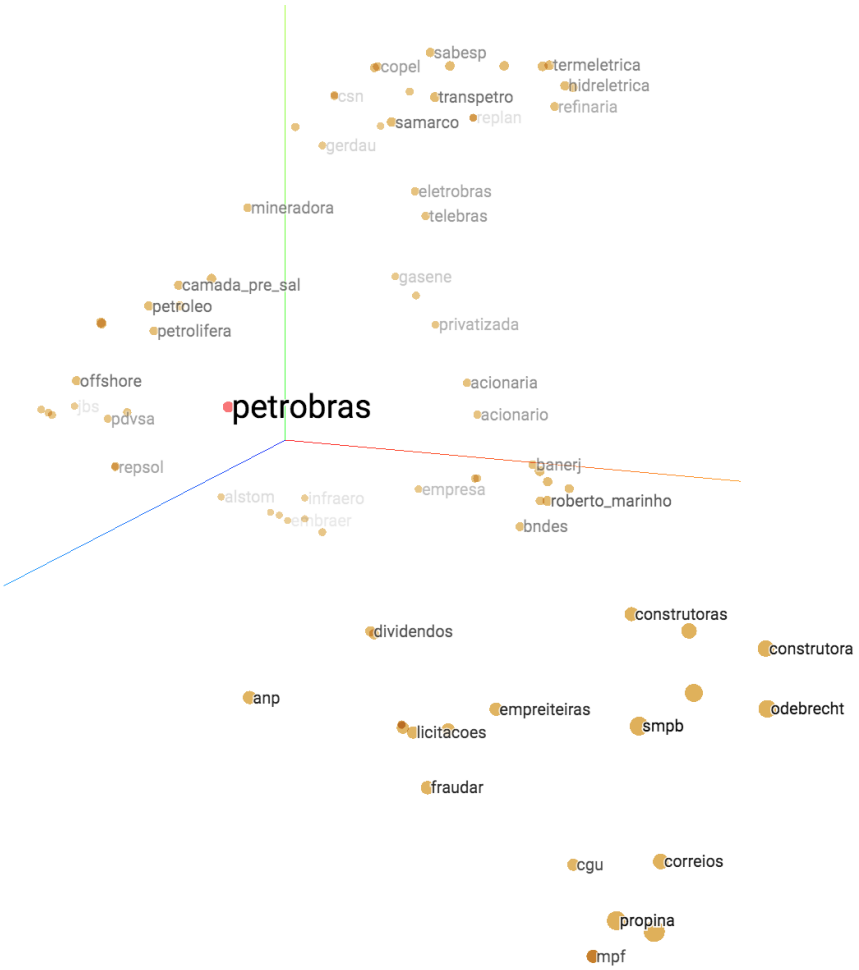


Figure 4.1: Petrobrás nearest neighbors.

4.3 Classifier Construction

We tackle the problem of time series forecasting as a classification problem of predicting if the value in the timestep t_{i+1} is higher, lower or equal with regard to the timestep t. The dataset can be seen as a collection of time series with minute granularity, each representing a day.

4.3.1 Preprocessing

The target variable of the problem at hand is the *open* price for each minute, from which we derive the label for each entry of the dataset, i.e., the label 0 represents a negative variation in the next minute, 1 a positive variation and 2 no variation. The smaller variation observed is of 1 cent of Brazilian Reais (BRL).

For example, some basic statistics of the PETR4 dataset is shown in Figure 4.2. In the left, we show the target variable time series and the dotted line is the point where the dataset is splitted into train and test sets. The train set corresponds to 90% of the dataset - 162 days or 77.142 minutes - and the test set 10% - 18 days or 8.512 minutes. In the right, we show that the class distribution is quite balanced in both train and test set. It can be noticed that the label *neutral* the dominant class by a small margin, which should not be a problem for a learning algorithm.

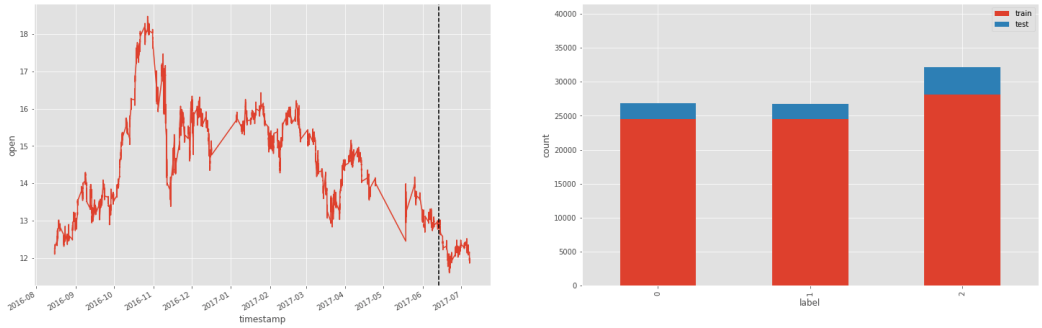


Figure 4.2: PETR4 stock target variable train test split and class distribution.

The dataset is composed of a total of 10 features, of which 8 are numeric and 2 text. The numeric features were normalized with zero mean and unit variance, also called standardization. For the text features, we computed the average of each word embedding, resulting in a 100 size continuous vector for entities and articles. For the trending stories entities, we computed the embedding by summing the target word with its context words. This approach adds first-order similarity terms and has been proven to be a better representation of a word than the single word embedding. Also, for each timestep, the features represent a window of 15 minutes.

4.3.2 Learning

The chosen machine learning algorithms to learn the task of predicting if the price in the next minute will have a positive, negative or neutral variation were Support Vector Machines (SVM) and Long Short Term Memory (LSTM) Networks.

5
Experimental Evaluation

In this chapter we present the results of the proposed models and how the introduced features - trending stories entities and most related articles titles - perform in comparison with the prediction using only the time series data.

The experiments were evaluated on the top 3 B3 stocks. Below, in Figures 5.1, 5.2 and 5.3, we show the class distribution for PETR4, ABEV3 and ITSA4 datasets. It can be noticed that the label *neutral* is predominant in all datasets, but quite unbalanced on ITSA4. The impact of an unbalanced class distribution is significant and can make the problem too difficult to learn, concentrating the predictions around the predominant class. We used the same machine to train all models, which was a Intel(R) Core(TM) i7-5960X CPU @ 3.00GHz, 64GB RAM and a Nvidia Tesla K40c GPU.

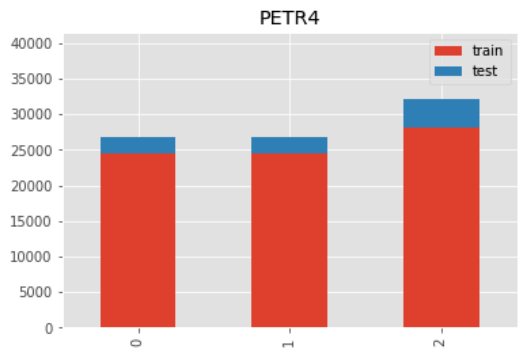


Figure 5.1: PETR4 class distribution.

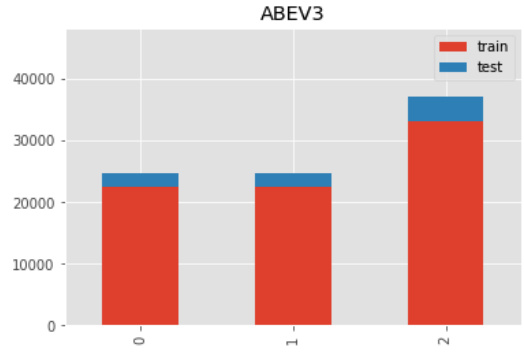


Figure 5.2: ABEV3 class distribution.

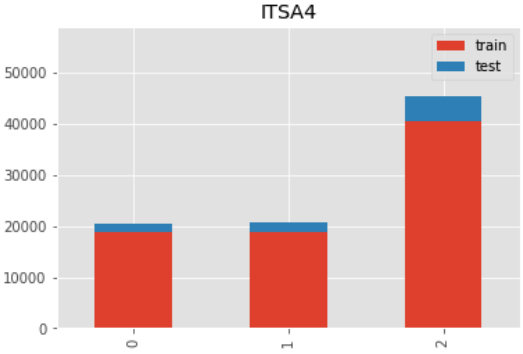


Figure 5.3: ITSA4 class distribution

5.1 SVM

The baseline chosen for the current prediction task was a Linear Support Vector Classification. This algorithm is implemented in the Scikit-learn library (48). Scikit-learn is largely written in Python, with some core algorithms written in Cython, which is a Python to C compiler, to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM (49); logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR (50). Below, in Table 5.1, we show the achieved accuracy in both train and test sets. The best results are highlighted with bold face.

Symbol	Features	Train Accuracy	Test Accuracy
PETR4	Numeric	69.27%	65.16%
	Numeric and Trends	69.53%	**66.42%**
ABEV3	Numeric	64.32%	63.65%
	Numeric and Trends	65.33%	**65%**
ITSA4	Numeric	67.02%	65.36%
	Numeric and Trends	68.37%	**66.78%**

Table 5.1: SVM results

The training times were approximately 3 minutes for the numeric features model and 4 minutes for the numeric and trends features model. The results shown are the mean of 3 experiment executions. Despite the small improvements in accuracy, the trends features resulted in better results for all experiments, proving that it contains valuable information. However, it was a surprise how well we were able to predict trends with only the time series data.

5.2 LSTM

In the last few years, there have been incredible success applying LSTMs to a variety of problems: speech recognition, language modeling, translation, image captioning, time series forecasting and so on. LSTM models are powerful enough to learn the most important past behaviors and understand whether or not those past behaviors are important features in making future predictions. Hence, after we set a baseline for the problem using SVM, we trained a LSTM network using Keras (45), which is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. We used TensorFlow (47) as the backend, since it is the most established software library for numerical computation using data flow graphs. As matter of fact,

in 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library.

Symbol	Features	Train Accuracy	Test Accuracy
PETR4	Numeric	71.30%	67.46%
	Numeric and Trends	71.66%	**69.24%**
ABEV3	Numeric	67.89%	66.57%
	Numeric and Trends	68.09%	**67.42%**
ITSA4	Numeric	69%	69.36%
	Numeric and Trends	69.07%	**69.66%**

Table 5.2: LSTM results.

The experiments were executed in approximately 8h or 500 epochs. We can notice that the baseline result was improved by a considerable margin, which may be attributed to LSTM capability of learning long-term dependencies. The architecture was a single layer of LSTM cells of size 32, then a dense layer (output layer) with softmax activations is used to get the output probabilities. We trained the network using batches of 4 sequences.

To extend this analysis, Figure 5.2 shows precision and recall for each class in both models trained on PETR4 data.

Symbol	Features	Metric	Class	Train Set	Test Set
PETR4	Numeric	Precision	negative	78%	66%
			positive	78%	68%
			neutral	61%	**68%**
		Recall	negative	75%	**75%**
			positive	73%	**70%**
			neutral	66%	61%
	Trends	Precision	negative	78%	**80%**
			positive	77%	**78%**
			neutral	62%	63%
		Recall	negative	76%	59%
			positive	74%	55%
			neutral	65%	**84%**

Table 5.3: LSTM evaluation metrics.

The results highlighted with bold face reflects the best metric found for a specific class in the test set evaluation. It can be noticed that with the trends features, the recall drops significantly for the classes *negative* and *positive*, which is compensated by a increase in precision. A higher precision in those classes means that we can make better decisions when the classifier outputs that the price will go up or down in the next minute. However, it makes more mistakes predicting neutral variations. Below we present the confusion matrix for the models.

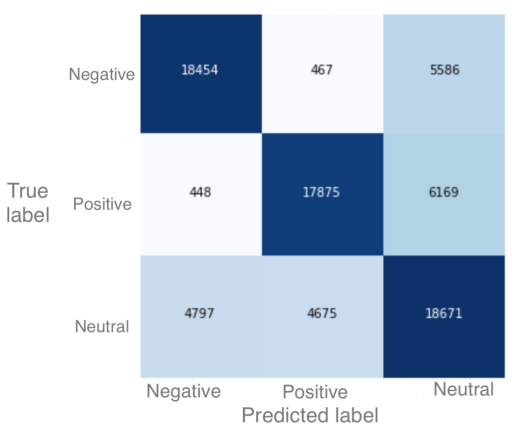


Figure 5.4: PETR4 Numeric features model train set confusion matrix.

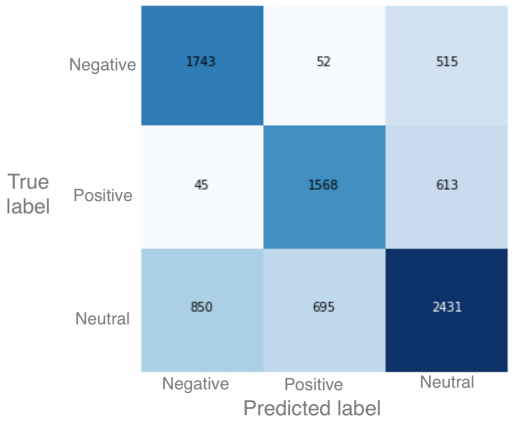


Figure 5.5: PETR4 Numeric features model test set confusion matrix.

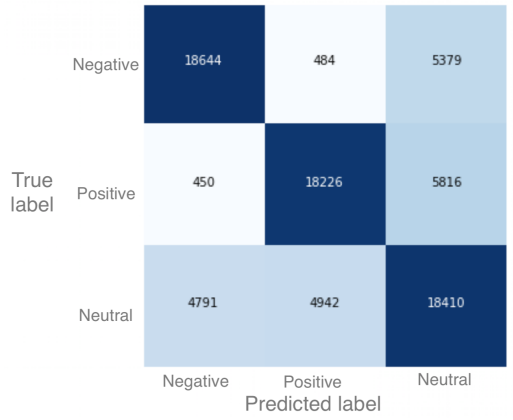


Figure 5.6: PETR4 Trends features model train set confusion matrix.

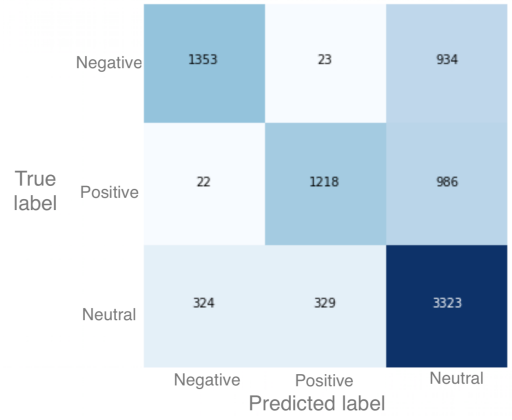


Figure 5.7: PETR4 Trends features model test set confusion matrix.

The results presented in Table 5.2 are reflected in the confusion matrices presented above. Comparing Figure 5.5 and Figure 5.7, we can note that the trends features model makes half the mistakes with regard to classes *negative* and *positive*, which means that is less likely that the classifier will predict class *negative* instead of *positive* and vice-versa. Also, the errors predicting class *negative* or *positive* when the correct class was *neutral* drops significantly with the trends model. However, the total number of correct predictions of class *negative* and *positive* were considerably reduced.

We analyzed in a similar way the results for the ABEV3 and ITSA4 stocks. However, the performance gain was not significant in order to draw conclusions about the relevance of the trends features. We suggest that only some stocks may be sensible to trending stories.

In summary, our results are consistent with the suggestion that during the period we investigate, Google Trends data may have provided some insight into future trends of the stock market. We observe a slightly increase in accuracy of our trend predictor model and the presented results suggest that these predictions can be exploited in the construction of profitable trading strategies.

6
Conclusions

In this final chapter, we conclude by describing the progress made towards the task of predicting trends in the stock market. Also, we suggest some future research directions that could provide the next steps along the path to a practical and widely applicable stock market predictor.

The initial objective of this work is that it might be viewed as a “first step" towards a trading algorithm for B3 stocks. With a reliable prediction for price fluctuations, we can get some edge in the market, although developing trading strategies based on the result of this work is as challenging as the prediction itself.

In summary, our results are consistent with the suggestion that during the period we investigate, Google Trends data did not only reflect aspects of the current state of the economy, but may have also provided some insight into future trends of the stock market. Using historic data from the period between August 2016 and July 2017, we observe a slightly increase in accuracy of our trend predictor model. Our results suggest that these predictions can be exploited in the construction of profitable trading strategies.

The quality of the proposed methodology has been thoroughly tested on a reasonable amount of time and it has been shown that the results that we have obtained are satisfactory, thus proving the validity of the idea of exploiting trending stories as a fundamental variable.

Even though we tried our best to provide a complete analysis of the stock prediction methodology based on Google Trends, it will be of sure interest to delve into many aspects of the methodology. The first thing will be to apply the same methodology to different sectors such as currency exchange. Another aspect that may deserve attention could be that of considering the ranking of trending stories. Moreover, the impact of extending the time lag of the prediction could be explored, in such a way to infer something on the possible long lasting effect of the trending topic.

We strongly believe that the idea of exploiting the trending stories of Google Trends for the prediction of price fluctuations of financial instruments is of sure practical interest even though it requires remarkable efforts. We conclude that these results further illustrate the exciting possibilities offered by

new big data sets to advance our understanding of complex collective behavior in our society.

This work has opened lots of branches for future research. The trending stories were labeled with the business category by Google, but a more fine-grained classification can help cleaning and result in selecting only trending stories that relate specifically to economy. Also, trending stories remain through many minutes in the rank list, which means that we can maybe discard some of them after we detected that the its search volume has stabilized after trending. This could be achieved by collecting the "Interest Over Time" data of stories.

The prediction were based on a sliding window of 15 minutes and a time lag of 1 minute. These variables can and must be extensively tested to find the most suitable lengths. Maybe a more coarse-grained prediction of 2 minutes or 5 minutes lag can give better results. Ideally, we should compute this lag by looking at how much the trending stories change over time, i.e., shifting of ranks and/or appearance of new stories.

The experiments were focused on proving the value of the trending stories data and consider model architecture exploration and hyperparameters optimization out of scope, which may provide crucial improvements in the final trend prediction model accuracy.

Finally, for each stock we need to train its own predictor if we want to trade the majority of B3 stocks. This would be too expensive to train in terms of time and computer processing. However, an approach to create one single and generic classifier can be explored to make the task of developing trade strategies using the type of predictor presented in this dissertation more viable.

Bibliography

[1] FAMA, EUGENE F. **The behavior of stock-market prices**. The journal of Business, 38(1):34–105, 1965.

[2] EUGENE F. FAMA. **Random walks in stock-market prices**. Financial Analysts Journal, 21:55–59, 1965.

[4] FAMA, E.. **Efficient capital markets: A review of theory and empirical work**. Journal of Finance, 25:383–417, 1970.

[5] FAMA, E.. **Efficient capital markets: Ii**. Journal of Finance, 46(5):1575–617, 1991.

[6] ALEXANDER, S. S.. **Price movements in speculative markets: Trends or random walks**. Industrial Management Review, 2:7–26, 1961.

[7] JENSEN, M.. **Some anomalous evidence regarding market efficiency**. Journal of Financial Economics, 6(2-3):95–101, 1978.

[8] QIAN, B.; RASHEED, K.. **Stock market prediction with multiple classifiers**. Applied Intelligence, 26(1):25–33, Feb 2007.

[9] BOLLEN, J.; MAO, H. ; ZENG, X.. **Twitter mood predicts the stock market**. CoRR, abs/1010.3003, 2010.

[10] JIA, H.. **Investigation into the effectiveness of long short term memory networks for stock price prediction**. CoRR, abs/1603.07893, 2016.

[11] PAGOLU, V. S.; CHALLA, K. N. R.; PANDA, G. ; MAJHI, B.. **Sentiment analysis of twitter data for predicting stock market movements**. CoRR, abs/1610.09225, 2016.

[12] DING, X.; ZHANG, Y.; LIU, T. ; DUAN, J.. **Using structured events to predict stock price movement: An empirical investigation**. In: EMNLP, p. 1415–1425. ACL, 2014.

[13] DING, X.; ZHANG, Y.; LIU, T. ; DUAN, J.. **Deep learning for event-driven stock prediction**. In: IJCAI, p. 2327–2333. AAAI Press, 2015.

[14] R. VARIAN, H.; CHOI, H.. **Predicting the present with google trends**. 88, 04 2009.

[15] GINSBERG, J.; MOHEBBI, M.; PATEL, R.; BRAMMER, L.; SMOLINSKI, M. ; BRILLIANT, L.. **Detecting influenza epidemics using search engine query data**. Nature, 457:1012–1014, 2009. doi:10.1038/nature07634.

[16] PREIS, T.; MOAT, H. S. ; STANLEY, H. E.. **Quantifying trading behavior in financial markets using google trends**. Sci. Rep., 3, Apr. 2013.

[17] FILHO, H. P. B.. **Classificação de sentimento para notícias sobre a petrobras no mercado financeiro**. Master's thesis, PUC-Rio, 2011.

[18] FILHO, H. P. B.. **Predição do comportamento do mercado financeiro utilizando notícias em português**. Master's thesis, PUC-Rio, 2014.

[19] **A proposal for the dartmouth summer research project on artificial intelligence**. `http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html`. Último acesso em 05/01/2017.

[20] RUSSEL, S. J.; NORVIG, P.. **Artificial Intelligence A Modern Approach**. Prentice Hall, first edition, 1995.

[21] DOMINGOS, P.. **A few useful things to know about machine learning**. Commun. ACM, 55(10):78–87, Oct. 2012.

[22] **Developing and adopting big data machine learning practice**. `http://bigdataknowhow.weebly.com/blog/archives/07-2014`. Último acesso em 05/01/2015.

[23] SIMPSON, P. K.. **Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations**. Pergamon Pr, 1990.

[24] JACKSON, T.; BEALE, R.. **Neural Computing: An Introduction**. Adam Hilger, 1990.

[25] MCCULLOCH, W. S.; PITTS, W.. **A Logical Calculus of the Ideas Immanent in Nervous Activity**. Bulletin of Mathematical Biology, 1943.

[26] MINSKY, M.; PAPERT, S.. **Perceptrons: An Introduction to Computational Geometry**. MIT Press, Cambridge, MA, USA, 1969.

[27] HEBB, D. O.. **The Organization of Behavior**. John Wiley, 1949.

[28] ROSENBLATT, F.. **The Perceptron: A Probabilistic Model For Information Storage and Organization in the Brain**. Psychological Review, 1958.

[30] RUMELHART, D. E.; HINTON, G. E. ; WILLIAMS, R. J.. **Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1**. chapter Learning Internal Representations by Error Propagation. MIT Press, Cambridge, MA, USA, 1986.

[31] HINTON, G. E.; OSINDERO, S. ; TEH, Y.-W.. **A fast learning algorithm for deep belief nets**. Neural Computation, 2006.

[32] **The dawn of big data**. `http://www-935.ibm.com/services/uk/en/attachments/pdf/IBM_BOA_Big_Data_General_WEB_v2.pdf`. Último acesso em 02/11/2014.

[33] HARRIS, Z.. **Distributional structure**. Word, 10(23):146–162, 1954.

[34] HOCHREITER, S.; SCHMIDHUBER, J.. **Long short-term memory**. 9:1735–80, 12 1997.

[35] GERS, F. A.; SCHMIDHUBER, J. ; CUMMINS, F.. **Learning to forget: Continual prediction with lstm**. Neural Computation, 12:2451–2471, 1999.

[36] KARPATHY, A.. **Connecting Images and Natural Language**. PhD thesis, Stanford University, 8 2016.

[37] HOCHREITER, S.. **Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München**, 1991.

[38] BENGIO, Y.; SIMARD, P. ; FRASCONI, P.. **Learning long-term dependencies with gradient descent is difficult**. Trans. Neur. Netw., 5(2):157–166, Mar. 1994.

[39] BENGIO, Y.; DUCHARME, R.; VINCENT, P. ; JANVIN, C.. **A neural probabilistic language model**. J. Mach. Learn. Res., 3:1137–1155, Mar. 2003.

[40] LE, Q. V.; MIKOLOV, T.. **Distributed representations of sentences and documents**. CoRR, abs/1405.4053, 2014.

[41] MIKOLOV, T.; CHEN, K.; CORRADO, G. ; DEAN, J.. **Efficient estimation of word representations in vector space**. CoRR, abs/1301.3781, 2013.

[42] RONG, X.. **word2vec parameter learning explained**. CoRR, abs/1411.2738, 2014.

[43] TOMAS MIKOLOV, WEN-TAU YIH, G. Z.. **Linguistic regularities in continuous space word representations**. Association for Computational Linguistics, May 2013.

[44] MIKOLOV, T.; LE, Q. V. ; SUTSKEVER, I.. **Exploiting similarities among languages for machine translation**. CoRR, abs/1309.4168, 2013.

[45] CHOLLET, F.; OTHERS. **Keras**. `https://github.com/keras-team/keras`, 2015.

[46] ŘEHŮŘEK, R.; SOJKA, P.. **Software Framework for Topic Modelling with Large Corpora**. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, p. 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[47] ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIÉGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y. ; ZHENG, X.. **TensorFlow: Large-scale machine learning on heterogeneous systems**, 2015. Software available from tensorflow.org.

[48] PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M. ; DUCHESNAY, E.. **Scikit-learn: Machine learning in Python**. Journal of Machine Learning Research, 12:2825–2830, 2011.

[49] CHANG, C.-C.; LIN, C.-J.. **LIBSVM: A library for support vector machines**. ACM Transactions on Intelligent Systems and Technology,

2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[50] FAN, R.-E.; CHANG, K.-W.; HSIEH, C.-J.; WANG, X.-R. ; LIN, C.-J.. **LIBLINEAR: A library for large linear classification**. Journal of Machine Learning Research, 9:1871–1874, 2008.

[51] DAMIEN, C.; AHMED, B. H. A.. **Predicting financial markets with Google Trends and not so random keywords**. Working Papers hal-00851637, HAL, Aug. 2013.

[52] HU, H.; TANG, L.; ZHANG, S. ; WANG, H.. **Predicting the direction of stock markets using optimized neural networks with google trends**. Neurocomputing, 285:188 – 195, 2018.