INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Deep Learning para BigData

FILIPE JOSÉ RIBEIRO CORREIA
Outubro de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Deep Learning para BigData

Filipe José Ribeiro Correia

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Profª. Ana Maria Madureira

Co-orientador: Prof. Jorge Bernardino

Porto, Outubro 2021

Dedicatória

“What seems unreasonable, unrealistic and impossible today can turn out to be inevitable tomorrow. It’s time for a new realism. It’s time for a new view of humankind.”
— Rutger Bregman, Humankind: A Hopeful History

Abstract

We live in a world where data is becoming increasingly valuable and increasingly abundant in volume. Every company produces data, be it from sales, sensors, and various other sources. Since the dawn of the smartphone, virtually every person in the world is connected to the internet and contributes to data generation. Social networks are big contributors to this Big Data boom. How do we extract insight from such a rich data environment? Is Deep Learning capable of circumventing Big Data's challenges? This is what we intend to understand. To reach a conclusion, Social Network data is used as a case study for predicting sentiment changes in the Stock Market. The objective of this dissertation is to develop a computational study and analyse its performance. The outputs will contribute to understand Deep Learning's usage with Big Data and how it acts in Sentiment analysis.

Keywords: Deep Learning, Big Data, Neural Network, Stock Data, Financial Markets, Social Networks

Resumo

Vivemos num mundo onde dados são cada vez mais valiosos e abundantes. Todas as empresas produzem dados, sejam eles provenientes de valores de vendas, parâmetros de sensores bem como de outras diversas fontes. Desde que os smartphones se tornaram pessoais, o mundo tornou-se mais conectado, já que virtualmente todas as pessoas passaram a ter a internet na ponta dos dedos. Esta explosão tecnológica foi acompanhada por uma explosão de dados. As redes sociais têm um grande contributo para a quantidade de dados produzida. Mas como se analisam estes dados? Será que Deep Learning poderá dar a volta aos desafios que Big Data traz inerentemente? É isso se pretende perceber. Para chegar a uma conclusão, foi utilizado um caso de estudo de redes sociais para previsão de alterações nas ações de mercados financeiros relacionadas com as opiniões dos utilizadores destas. O objetivo desta dissertação é o desenvolvimento de um estudo computacional e a análise da sua performance. Os resultados contribuirão para entender o uso de Deep Learning com Big Data, com especial foco em análise de sentimento.

The objective of this dissertation is to develop a computational study and analyse its performance. The outputs will contribute to understand Deep Learning's usage with Big Data and how it acts in Sentiment analysis.

Palavras-chave: Deep Learning, Big Data, Redes Neuronais, Dados de ações, Mercados Financeiros, Redes Sociais

Agradecimentos

Começo por agradecer à minha família pelo apoio. Aos meus pais pela ajuda que foram e são no dia a dia. À minha namorada Silvana pela paciência e ajuda, bem como pelo encorajamento.

Agradeço também ao Instituto Superior de Engenharia do Porto que me acolheu durante 6 anos, tanto na licenciatura como no mestrado. Aos professores desta instituição que me ensinaram o que sei e que me permitiu escrever esta tese. Mais especificamente, agradeço aos orientadores deste projeto, a Professora Ana e o Professor Jorge pela disponibilidade e acessibilidade que demonstraram.

Agradeço aos meus colegas da Farfetch que me ensinaram também o que sabiam de dados e contribuíram muito para que fosse capaz de chegar ao fim desta tese.

Por fim agradeço a todos os que me apoiaram e acreditaram em mim.

Table of Contents

Figure List

Table List

Acronyms and Symbols

Acronym List

ACID	*Atomicy, Consistencty, Isolation, Durability*
AC-BiLSTM	*Bidirectional LSTM with Attention Mechanism and Convolutional Layer*
AI	*Artificial Inteligence*
AM	*Application Master*
ANN	*Artificial Neural Network*
API	*Application Programming Interface*
ARC	*Recurrent Convolutional Neural Network with Attention for Sentiment Classification*
BDA	*Big Data Analysis*
CNN	*Convolutional Neural Network*
CUDA	*Compute Unified Device Architecture*
DAG	*Directed Acyclic Graph*
DFS	*Distributed File System*
DNN	*Deep Neural Network*
ETL	*Extract, Transfer, Load*
GPGPU	*General Purpose Graphical Processing Unit*
GPU	*Graphical Processing Unit*
GRU	*Gated Recurrent Unit*
HAN	*Hierarchical Attention Network*
HDFS	*Hadoop Distributed File System*
IoT	*Internet of Things*
IWV	*Improved Word Vectors*
LSTM	*Long Short-Term Memory*

ML	*Machine Learning*
MPI	*Message Passing Interface*
nCV	*Nested Cross Validation*
NLP	*Natural Language Processing*
NN	*Neural Net*
OLTP	*Online Transactional Processing*
PoS	*Parts of Speech*
ReLU	*Rectified Linear Unit*
RDD	*Resilient Distributed Dataset*
RDBMS	*Relational Data Base Management System*
RM	*Resource Manager*
RNN	*Recurrent Neural Network*
RvNN	*Recursive Neural Network*
SS-BED	*Sentiment Specific Word Embedding*
SVM	*Support Vector Machine*
TanH	*Hyperbolic Tangent*
YARN	*Yet Another Resource Negotiator*

1 Introduction

Big Data is considered to be valuable and capable of unleashing new organizational capabilities. Clickstreams from the web, social media contents, video data, call centre voice data, genomic and proteomic data from biological and medical research are among some sources. In the past decade, the volume of data has immensely increased, along with the development of data-generating, extracting, addressing, storage and output technologies (Wang and Wang, 2020).

Deep Learning is an approach to Artificial Intelligence (AI), more specifically, a type of Machine Learning (ML), which allows system improvement through data insights and the experience derived from them (Lecun, Bengio and Hinton, 2015). ML is where algorithmic skills of data science and statistical analysis meet, and the result is a set of approaches to inference and exploration which are mostly about effective computation (Cleary, 2019).

The current chapter presents the problem, context, motivation, method, contribution, and this paper's structure.

1.1 Problem

A quick and generic way to define the concept of Big Data is: "Significantly large datasets beyond the ability of traditional database software to store, capture, manage or analyse". Going deeper into the subject, Big Data is a result of an increasingly higher data generation and new database technologies. Internet of Things (IoT) devices, such as smartphones, tablets, wearables, etc. have been adopted worldwide by consumers and industries which allowed the access to the internet by their users. The International Data Corporation estimates that in 2025 there will be 175 Zettabytes of data, increased from the 2 Zettabytes existing in 2010 (Wang and Wang, 2020). To use this data, specialized tools and methods have been developed and professionals have specialized in its collection, management and analysis.

Data analysis is one purpose of ML algorithms, which extract meaning from the data. ML algorithms have never been so employed, but never have they also been so challenged such as by Big Data in obtaining knowledge. Big Data offers massive volumes of data and information for ML algorithms to work upon, allowing the extraction of patterns or building analytical models. Massive amounts of highly variating data coupled with the speed of which it is generated and needs to be processed make it a challenge to use with traditional ML algorithms (Zhou *et al.*, 2017). Traditional ML algorithms are not able to handle the verticality or the dimensionality that Big Data offers in a timely manner (L'Heureux *et al.*, 2017) and are therefore not able to scale as necessary to fully uncover the hidden value of Big Data. Neural Networks (NNs) are a possible solution for the problems of traditional ML algorithms. By building Deep Neural Networks (DNNs) it is possible to increase the predictive power of a NN. In a DNN, some neurons are activated for specific data points, whereas others are more global, theoretically, allowing for a high level and low level analysis of Big Data (Samek *et al.*, 2021).

1.2 Context

There is a growing academic effort to find ways to take advantage of the vast information capabilities provided by Big Data using ML to gather insights. Traditional ML algorithms are gradually slower at analysing big datasets, providing good analysis at a time when it is no longer needed, or not producing analysis at all. There are papers like Zhou *et al.* (2017), L'Heureux *et al.* (2017), Jeong, Hassan and Sangaiah (2019), Xie *et al.* (2018) and Roh, Heo and Whang (2018) which study some challenges, approaches and opportunities of working with Big Data. These authors provide a comparison between a variety of ways to deal with the challenges presented. Among them are some optimization techniques adaptable to some traditional ML algorithms plus pioneering paradigms which are gaining support within the academic community.

Deep Learning is appointed as one of the possible ways to tackle some issues presented by Big Data, mainly feature engineering, non-linearity, data heterogeneity, uncertain, dirty and noisy data (L'Heureux *et al.*, 2017; Zhou *et al.*, 2017). Deep Learning takes advantage of multiple processing layers to learn representations of data with multiple levels of abstraction (Lecun, Bengio and Hinton, 2015; Gao and Wang, 2019). It is also capable of finding complex structures present on data by using the backpropagation algorithm to indicate how the network should readjust its internal parameters that are used to calculate the representation in each layer from the representation in the previous layer (Lecun, Bengio and Hinton, 2015).

Along with new paradigms for Big Data analysis, new technologies and methodologies of storage are being adopted, since traditional databases fall short on keeping up on the scalability necessary to handle this type of data (Madden, 2012) and suffer especially with data streaming. Inoubli *et al.* (2018) offers a look on some scalable, fault tolerant and efficient frameworks pointed directly at Big Data and include frameworks like Hadoop or Spark which drop the Atomicity, Consistency, Isolation, Durability (ACID) principles (Haerder and Reuter, 1983) of traditional transaction-oriented databases in favour of processing performance.

1.3 Objective

The focus of this dissertation is on investigating, developing, and implementing a system of automatic classification, based on Deep Learning, which allows, in each scenario, to validate its adequacy and efficiency as a solution. The development of a computational study with the intention of analysing the performance of the prototype, by using statistical techniques on the achieved outcomes, is considered vital. The developed system will be acting upon social network data as it fits within the description of Big Data and the classification will be done by correlating the sentiment of social network users to the fluctuations of financial indicators. The prediction of stock value increments or decrements is not the final goal of this thesis, but the means which will allow the comprehension of whether Deep Learning is a good way to process and analyse Big Data.

Parallel processing is part of the scope of this dissertation and will be achieved by using a GPU. This will also increase the processing capacity or power, which will help processing a big volume of information, allowing for an increment on efficiency.

1.4 Method

To be able to investigate, develop and implement an automatic classification system using Deep Learning over Big Data, a data source (or many) is needed, and the data which it produces must be compliant with the characteristics previously defined as Big Data characteristics. Financial markets which are soaring in data production can take advantage of data models to perform precision marketing (Pang and Liu, 2020) and perform stock value prediction. This will be the core theme for data analysis, and it will be performed upon data streams from social networks (like Twitter or Reddit) which will then be compared to the actual values for information on the system performance.

Social network data is usually not structured, consisting of the words of its users (Al-Garadi *et al.*, 2019). In order to use this kind of data to feed a Deep Learning algorithm, it must be pre-processed, mainly through the use of a Natural Language Processing (NLP) algorithm (Bollen, Mao and Zeng, 2011; Bordes *et al.*, 2011; Gui, 2019) or using Deep Learning Neural Nets (NN) for feature extraction. After this, the same network may be able to extract insights from the data it previously extracted features from.

All this data must first be stored, go through a phase of data preparation and then it may be used in the NN. The way this is going to be achieved is through a data pipeline which will perform Extraction, Transform and Load (ETL) operations, ultimately leading data into the NN. A suitable data warehouse technology is a must in this solution, as well as an appropriate framework for the implementation of a Deep Learning algorithm. The choice of technologies is an important aspect of this investigation.

1.5 Contribution

The main contributions of this dissertation are the following:

- Usage of Deep Learning for feature extraction from unstructured data.
- Combining different methods of Deep Neural Networks for Natural Language Processing and Sentiment Analysis to infer variations on Stock Data Value.
- Understand how to process Big Data efficiently using Deep Learning and specialized hardware techniques.
- Stock market prediction, which has already been used by with social network data, but not using a Deep Learner.
- Big Data extraction, management, and analysis strategies.
- Understand how different techniques of Deep Neural Networks act on sentence data.
- How numerical data can enrich the techniques used for sentiment prediction.
- Understand how to extract, transfer, load, store and pre-process Big Data.

1.6 Document Structure

The rest of this dissertation is structured as follows. Chapter 2 consists of the state of the art, presenting a broad view of how the themes presented in this chapter are being addressed in the state of the art. Chapter 3 contains the value analysis, which explains the value created by this work. Chapter 4 clarifies how the work was structured, which factors went into selecting the software to be used, selecting alternative architectures and data sources. Chapter 5 describes the implementation of what was explained in the previous chapter. Chapter 6 explains which experiments were made and how they were evaluated. Finally Chapter 7 presents the main conclusions of this dissertation and ideas for future work.

2 State of The Art

This chapter provides an in-detail overview of the state of the academic world, as of time of writing, regarding the main themes of this dissertation. The focus is on Big Data, an emerging concept which relates to data in extreme conditions when compared to traditional databases and datasets. Deep Learning, a method of collecting insights on data. Financial Data, which demonstrates the value of different companies in a stock market. And Social Network Data, which consists of big amounts of unstructured data generated in real time. Financial decisions have benefited from Big Data Analysis (BDA) models, proving their feasibility and impact on decision making (Pang and Liu, 2020). Social Networks continuously produce quintillion bytes of data daily (i.e.: Facebook stores 260 billion photographs in over 20 Petabytes of storage) which can provide very distinct information on user behaviour, opinions, sentiment, etc. (Al-Garadi *et al.*, 2019).

2.1 Big Data

The interest in this type of data has been increasing with various projects being launched to understand the opportunities and develop solutions. The USA was one of the leaders in this field, launching a Research and Development project in 2012 with countries such as Japan adopting it as an important aspect of its national strategy. The United Nations issued a report on the Opportunities and Challenges of Big Data for Development (Oussous *et al.*, 2018). Due to all this interest, many new technologies, frameworks and models were created with focus on areas such as Smart Grids (Stimmel, 2016), Health (Nambiar *et al.*, 2013), Internet of Things (Chen *et al.*, 2014), Political services and utilities as well as Transportation and logistics (Rajaraman, 2016).

Big Data has been characterized by “3 V's”, comprised of volume, variety and velocity (Zhou *et al.*, 2017) with two more “V’s” starting to be considered in the academic world, namely value and veracity:

- Volume: quantity or amount of data, which is usually higher than in traditional datasets
- Variety: In type, nature, format, etc. This data can be comprised of images, videos, phrases, numbers, etc. and come from different sources
- Velocity: the speed of data generation
- Value: the insights one can take from it and the impact these may cause
- Veracity: trustworthiness and quality of captured data

Table 1: Big Data aspects and challenges (L'Heureux et al., 2017) shows the link between the 5 Big Data "V's" and the current challenges they present.

Table 1: Big Data aspects and challenges (L'Heureux et al., 2017)

Aspect	Challenge
Volume	• Processing Performance • Curse of Modularity • Class Imbalance • Curse of Dimensionality • Feature Engineering • Non-Linearity • Bonferroni's Principle • Variance and Bias
Velocity	• Data availability • Real-Time Process/Streaming • Concept Drift • Independent and Identically Distributed Random Variables
Variety	• Data Locality • Data Heterogeneity • Dirty and Noisy Data
Veracity	• Data Provenance • Data Uncertainty • Dirty and Noisy Data
Value	• Context dependant

The amount of data provided by Big Data can result in highly dimensional data, which may need the use of Tensors, multi-dimensional arrays of numbers, which allow more complex intrinsic structures with higher-order data. For example, a colour image can be regarded as a 3-way tensor due to its three channels; a grey scale video can also be viewed as a 3-way tensor indexed by two spatial variables and one temporal variable. (Zhou *et al.*, 2018)

From the research and development projects, new technologies emerged for the different stages of Big Data, ranging from data extraction to data analytics. The technological layers in Figure 1 have been created by players like Apache which introduced whole Big Data ready environments, like Hadoop (Oussous *et al.*, 2018). The next subsections address these layers and the available solutions.

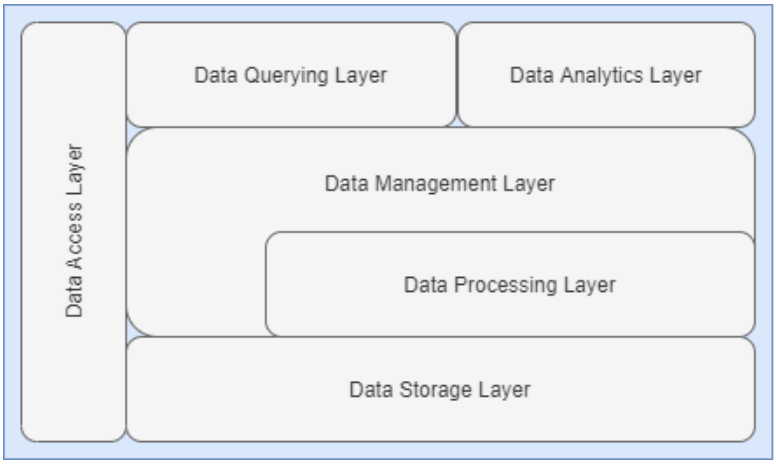


Figure 1: Big Data technological layers

2.1.1 Data Storage Layer

Big Data specialized storage technologies may end up sacrificing properties such as data consistency to maintain fast query response, dropping the traditional ACID principles of database properties (Haerder and Reuter, 1983). These technologies have much more power in handling unstructured and semi-structured data at a large scale. Table 2 presents an overview of current technologies and their properties based in the work of Cavanillas, Curry and Wahlster, 2016.

Table 2: Big Data Storage Layer

Storage System	Properties
Distributed File Systems	• Store large amounts of unstructured data in a reliable way in multiple machines • Well suited for quickly ingesting data • Well suited for bulk processing • Need a Big Data Querying layer to simplify data querying • Reached the level of de-facto standard for Big Data
NoSQL Databases	• Non-relational data models • Semi-structured and unstructured data • Columnar stores (Abadi, Boncz and Harizopoulos, 2009) • Data storage in graph structures, suitable for highly associative data (e.g.: social networks)
NewSQL Databases	• Comparable scalability to NoSQL • ACID support for transactions • Non locking concurrency mechanism • Uses SQL • Scale-out architecture with high node performance

Distributed File Systems (DFS) are the most prominent technology in terms of Big Data Storage thanks to some as Hadoop Distributed File System (HDFS), part of the Hadoop environment. Other alternatives which may be considered are HBase, also a part of the Hadoop Suite built on top of HDFS and is designed for low latency operations (Oussous *et al.*, 2018), Ori, a peer-to-peer DFS with opportunistic synchronization (Mashtizadeh *et al.*, 2013) and Coda, a file system for a large-scale distributed computing environment composed of Unix workstations (Satyanarayanan *et al.*, 1990).

NoSQL technologies have a presence in traditional data storage and have been rethought to fit in with Big Data standards. As opposed to traditional SQL databases, NoSQL ones are flexible, inherently supporting horizontal data partitioning across processors or even across data centres (Lourenço *et al.*, 2015).

There are different types of NoSQL databases, such as Key-value, Columnar, Graph, Document and Native Object.

Key-value databases, whose characteristics include real-time processing, horizontal scalability, reliability, and high availability. These systems store data as kay-value pairs or maps. Technologies like Redis, Apache Cassandra, Membase and Voldemort are examples of key-value NoSQL technologies (Gudivada, Rao and Raghavan, 2014).

NoSQL columnar databases are very similar to Relational Database Management Systems (RDBMS) as they designed to efficiently return entire rows of data. These databases are tolerant to temporary inconsistency, have flexible database schema, sparse data and high-speed insert and read operations. Technologies include Google Bigtable, Microsoft SQL Server, Apache HBase and Oracle RDBMS Columnar Expression (Gudivada, Rao and Raghavan, 2014).

Graph databases are NoSQL data models which can show static or dynamic relations between objects, just as in a graph. These are popular for social network data, as well as access control and authorization systems (Gudivada, Rao and Raghavan, 2014). Neo4j (Vukotic *et al.*, 2015), DGraph (Meng *et al.*, 2012) and JanusGraph, which is capable of integrating with Hadoop (Anikin, Borisenko and Nedumov, 2019; Šestak *et al.*, 2021) are examples of Graph databases.

Other NoSQL database models include Document databases like MongoDB, Native XML like BaseX or Sedna, and Native Object Databases such as WakandaDB (Gudivada, Rao and Raghavan, 2014).

NewSQL databases propose the ability to scale modern on-line transaction processing (OLTP) present in NoSQL while still maintaining the traditional model of operation of traditional RDBMS such as ACID guarantees. This enables the execution of concurrent transactions to ingest new information and modify the state of the database using SQL. Examples include VoltDB, Google Spanner and CockroachDB (Pavlo and Aslett, 2016).

2.1.2 Data Processing Layer

Specialized engines are commonly used for processing Big Data, given the data explosion, techniques like parallel processing are essential to manage a massive volume of data in a timely manner. There are already a lot of popular parallel processing models, including Message Passing Interface (MPI), which exploits multi-machine/multi-core infrastructure, General Purpose computing on the GPU (GPGPU), MapReduce and other MapReduce-like models (Ji *et al.*, 2012).

MPI takes advantage of multi-core computing, a paradigm where a single chip is used for two or more independent processors. Multi-core processing supports high scaling but has a high cost in manufacturing software. MPI also exploits multi-machine processing, which may be defined as a network of computers where resources like memory and data storage are share and is designed for high performance parallel processing (Dahiya, 2020). This model requires a big investment in hardware and will not be considered for this dissertation.

GPGPU uses, as the name indicates, a graphical processing unit (GPU), which is a programmable computer chip that can perform rapid calculations with a high degree of parallel processing. The main use of GPUs was to deal with the calculations involved in 3D models in computation (e.g.: 3D modelling, videogames)(Dahiya, 2020). These chips have been gaining popularity among researchers and developers. The Compute Unified Device Architecture (CUDA) is a C based programming language which resulted of this newfound popularity and is specifically designed for NVIDIA GPUs (Lee, Min and Eigenmann, 2009).

MapReduce is very popular as a Big Data processing model in both the industry and academia and has two major advantages: the model hides details related to data storage, distribution, replication, and load balancing, etc. and is very simple to implement, needing only the specification of two functions (map function and reduce function). The Map function divides the input data into individual data partitions that constitute key-value pairs, as depicted in Figure 2, which are sent into a Mapper that processes them one by one.

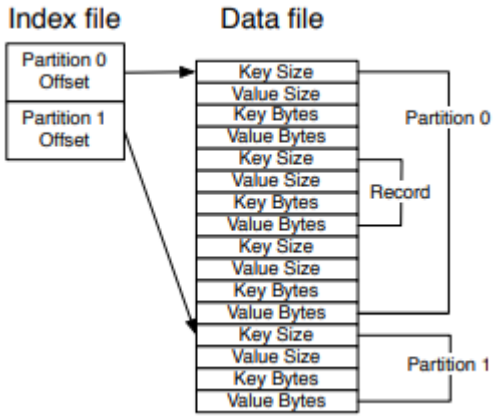


Figure 2: Map task index and data file format (Condie *et al.*, 2010)

Each data partition is assigned to a unique compute node. The framework must then collect all the key-value pairs, sort, and group them by key. The Reduce function is used to process the intermediate data. It aggregates the values associated to each key and produces the output key-value pairs. The final step is to store all key-value pairs in an output file (Oussous *et al.*, 2018).

Hadoop provides a MapReduce implementation to use with its HDFS file system as well as a more generic data processor, Yet Another Resource Negotiator (YARN). YARN provides better scalability, enhanced parallelism and advanced resource management compared to MapReduce by dropping MapReduce's monolithic implementation and separating resource management functions from the programming model.

Referenced in Figure 3 are the Resource Manager (RM) and Application Master (AM). The RM is a daemon which runs on a dedicated machine which tracks resource usage and node liveness, enforces allocation invariants, and arbitrates contention among tenants. The AM coordinates the logical plan of a single job by requesting resources from the RM, generating a physical plan from the resources it receives, and coordinating the execution of that plan around faults. In the context of Figure 3, MapReduce is just one of the applications running on top of YARN (Kumar Vavilapalli *et al.*, 2013). It also provides both batch processing and real-time interactive processing, and is compatible with MapReduce's Application Programming Interface (API) (Oussous *et al.*, 2018).

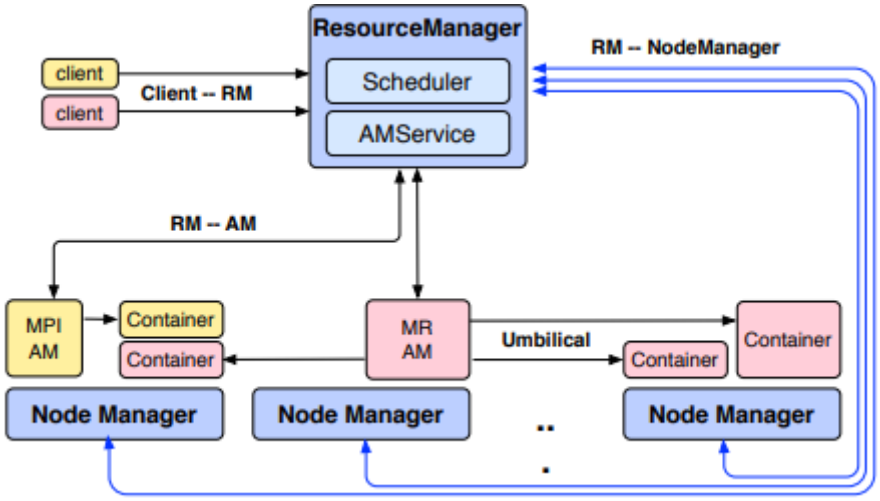


Figure 3: YARN architecture (Kumar Vavilapalli et al., 2013)

Spark and Storm are also processing engines, which are MapReduce-like models (Inoubli *et al.*, 2018), both with a high focus on real-time data processing. Spark targets mainly machine learning and interactive querying workloads. It is built to take advantage of resilient distributed datasets (RDD) to achieve performance improvements over classic MapReduce, and has been recently ported to YARN (Kumar Vavilapalli *et al.*, 2013). Storm is an open source distributed real-time processing engine which is mainly designed for parallel streaming and it is capable to scale across a machine cluster. Storm and MapReduce have been combined, using MapReduce as a batch processor and Storm as a stream processor. It may also be ported to YARN, unlocking more flexibility (Kumar Vavilapalli *et al.*, 2013).

2.1.3 Data Querying Layer

Data visualization is important for analysing, interpreting, and displaying data in significant ways (Ali *et al.*, 2016). Data visualization aids at finding interesting patterns and may even be way to debug data problems in a data pipeline. Visualization may help when tracking data provenance problems such as when data sources are not known (This relates with the Veracity component of Big Data).

When it comes to visual analysis, there are mainly two ways to do it:

- *Ad hoc* queries, queries which are entered by a user which may be used only once, it is a kind of query which is loosely typed when needed, for example, to find some specific data.
- Dashboards, which are typically graphical interfaces which can be programmed to run specific queries on a schedule, to show information on a structured way. These may have a backend built on top of typical *ad hoc* query languages such as SQL. These kinds of requests, when made in a traditional database are normally searching for a small amount of data, as opposed on when used with Big Data,

where there is a need to employ an optimized query engine to aid with the execution time (Rodrigues, Santos and Bernardino, 2017) .

Table 3 presents a comparison of Big Data querying engine features for some available. Hadoop, as in the previous layers, provides a data warehouse system for *ad hoc* querying, named Hive, which also provides table and storage management services, and relies heavily in map reduce, on which Impala was built upon, expanding Hive's key components, focusing on Interactive querying, as opposed to Hive's batch processing focus (Rodrigues, Santos and Bernardino, 2017) . HAWQ, or Hadoop With Query, as the name suggests is also built mainly for use with Hadoop. It has a complex layered architecture and relies on PostgreSQL to parse SQL syntax (Rodrigues, Santos and Bernardino, 2017). Drill was designed to query NoSQL and databases like HDFS, being able to access different data file formats (e.g.: CSV, JSON, PARQUET). Drill works by using "Drillbits" which are responsible for accepting requests, processing data and returning the output to the client (Rodrigues, Santos and Bernardino, 2017).

Table 3: Comparison of Big Data querying engines

Querying Tool	Language	Type of Language	Data structures	Schema	Data Access
Hive	HiveQL (SQL-like)	Declarative	Suited for structured data	has table metadata stored in the database	JDBC, ODBC
Drill	Calcite (Parses SQL)	Data flow	File-based data	distributed cache to manage metadata	Drillbit service
HAWQ	SQL	Declarative	Suited for structured data	has table metadata stored in the database	HAWQ engine
Impala	SQL (HiveQL extension)	Declarative	Suited for structured data	has table metadata stored in the database	JDBC, ODBC
Spark	Spark SQL	Resilient Distributed Datasets	Works with DataFrames[1]	Schema is optionally defined at runtime	Spark Core

[1] Equivalent to relational database tables

Spark is a multiuse tool and has been mentioned in section 2.1.2. As a processing tool, it is designed to support a large range of workloads, like batch and streaming processing, as well as interactive queries and iterative algorithms. This tool utilizes Resilient Distributed Datasets (RDD), which describes an unchangeable set of objects that are partitioned and scattered across multiple nodes of a cluster, allowing parallel processing (Rodrigues, Santos and Bernardino, 2017). Spark Core is the general execution engine for the platform, and it uses a master/worker architecture where a master node manages all workers executing tasks as illustrated in Figure 4.

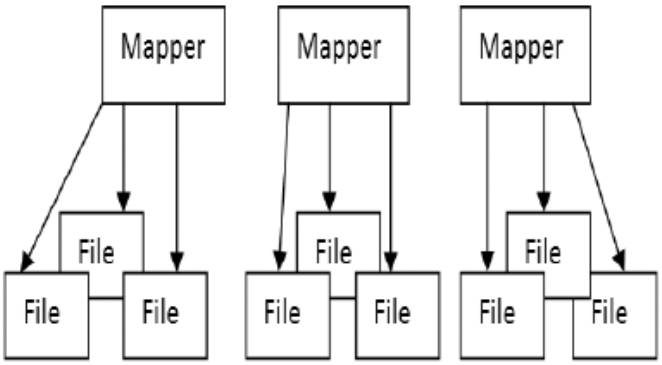


Figure 4: Spark Master/Worker topology (Verma, Mansuri and Jain, 2016)

2.1.4 Data Access Layer

A data pipeline has a starting point, where data is collected before being stored in a data warehouse. Collected data can have many distinct origins, it can originate from IoT devices, from sensors, APIs, social network data, etc. Companies like Facebook or Twitter produce an enormous amount of data from posts, tweets, comments, etc. which can provide value for companies on subjects like customer satisfaction. Clickstream data from webpages can provide insights to user behaviour. Sensor data, like car cameras, produce video data which can be used to train self-driving cars.

There are three main types of data:

- Structured data, which has a specific format and is stored into rows and columns, such as data stored in traditional databases;
- Unstructured data, which does not have any kind of structure associated, making it hard to analyse (e.g.: e-mail, audio, video);
- Semi-structured data, which lies between both previous data types, does not have any complex structure, but has specific information (e.g.: xml files) (Alwidian *et al.*, 2020).

Data Ingestion is the process which consists of moving data from different data sources into other systems for processing. Each business case may need a specific data tool for

ingestion. Parameters like data frequency, format, velocity, and others may influence the tool of choice. There are specialized tools for batch data ingestion, which consists of batches ingested at a defined time or data size interval, useful for offline non time sensitive data analysis. There are also tools specialized in data streaming, also known as real time data, which has a continuous and quick flow, fit for use cases which require real time data analysis like marketing recommendations.

Some data ingestion technologies are presented in Table 4 with their respective attributes. Sqoop is a tool which integrates well with Hadoop's HDFS and allows for bidirectional data transfers between relational databases and Hadoop, it is used by companies like Apollon Group and Coupon.com. Flume also integrates well with Hadoop; however, its data has a unidirectional flow into HDFS. Notable Flume users include companies such as Meebo, SimpleGeo and Sharethrough. Kafka is the only ingestion tool in the table which is not natively integrated with HDFS, although it is possible to program it to be. Kafka's notable users include Uber, Booking.com and Slack. NIFI stands out from the formerly mentioned tools as it is the only one which has as web interface and the only which is indicated for both batch and stream ingestion. NIFI is integrated with HDFS and has a bi-directional flow of data (into and out of HDFS). NIFI noteworthy customers include Macquarie Telecom Group, Hastings Group and Payoff (Alwidian *et al.*, 2020).

Table 4: Comparison of data ingestion tools (Oussous et al., 2018; Alwidian et al., 2020)

Ingestion Tool	Ingestion Type	Data Nature	Data Structure	Type of Loading
Sqoop	Batch Data	RDBMS	RDBMS schema	Not Event Driven
NIFI	Batch and Stream Data	Flow creation between different systems	Optional schema definition	Both
Flume	Stream Data	Continuously generating data sources	Schema can be obtained from the data itself	Event driven
Kafka	Stream Data	Messaging streaming data	Optional schema definition	Event driven
Storm - Spout	Stream Data	Continuously generating data sources	Optional schema definition; Schema may vary between input and output	Event driven
Spark	Batch and Stream Data	HDFS	Optional schema definition	Both

Storm and Spark, also included in Table 4, are not specifically designed for data ingestion, but have features which allow for data ingestion. Storm has an architecture based on a topology of spouts and bolts, as shown in Figure 5, where a spout is a source of data and a bolt is used to process input streams and produce outputs, which allows Storm to produce data transformations on streams. Storm can integrate synchronous and asynchronous data

sources (e.g.: Kafka, Shell) and supports any type of output system, be it relational or otherwise (Oussous *et al.*, 2018).

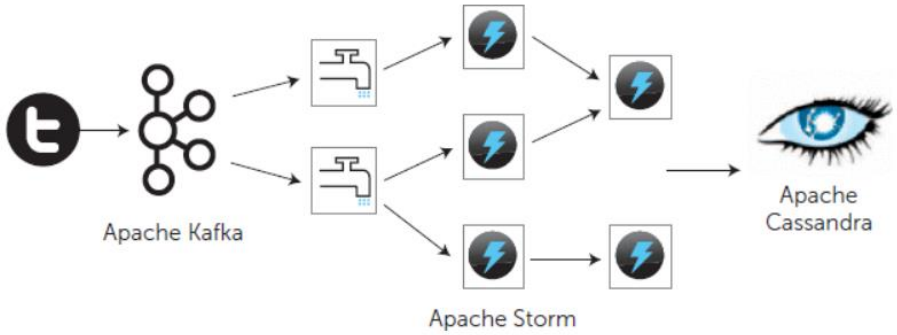


Figure 5: Example Storm use case (Ranjan, 2014)

Spark, which is itself a processing framework, is not usually used to stream data into HDFS but rather out of it, seeing its use in data analytics as a method of loading data into the tools which deal with its analysis, which makes it more of a data access tool than a data ingestion one (Oussous *et al.*, 2018).

2.1.5 Data Analytics Layer

Data analytics is comprised of various themes, from real-time data dashboards, advanced data visualization, to machine learning algorithms which provide insights through data classification and predictions, based on correlation (Russom and Org, 2011). Business Intelligence has seen adoption in companies where recommendation systems, which infer likes and preferences from others, are common as a use case (Tereso and Bernardino, 2011). Pattern mining finds interesting relations in data. Clustering partitions a large number of patterns according to their similarity (Londhe and Prasada Rao, 2018).

Use cases of data analysis include genetics and genealogy, which have been employing it to research how DNA sequences can affect disease susceptibility and resilience, to help cure or prevent these diseases by creating genetic trees of subjects. Healthcare employs data analysis on X-Ray, MRI, and other medical imaging to detect blood cloths, tumours and even organ delineation. Pharmaceutical companies employ simulations and mathematical models to aid in the creation and shorten development times of new drugs. Chatbots and virtual assistants use Natural Language Processing analysis to interact with users. The financial sector analyses data for fraud prevention, credit risk analysis and customer spending patterns. Agriculture depends on weather data analysis, soil quality or even sales data. Even other business use data analysis to give customers better experiences (e.g.: sales recommendations), and also predict sales (Ekka and Jayapandian, 2020) .

Dashboard technology like Tableau (Wen and Yang, 2020) and Power BI (Ferrari and Russo, 2016) allow a visual representation of analyses made from data through the use of graphs, bar plots, etc. These tools are suited for analytics which are repetitive and give overviews on data. Other tools like Mahout (Owen *et al.*, 2011), R studio (Oussous *et al.*, 2018), Tensorflow (Abadi *et al.*, 2016), Spark (Oussous *et al.*, 2018), etc. were created to make analytical models, using statistical algorithms, machine learning, and Deep Learning, which will be discussed further in section 2.2.

2.1.6 Management Layer

The last level of a Big Data technology stack is the management layer, which includes the storage, coordination, workflow, and system deployment. Storage management refers to the management of metadata related to data storage, allowing for interoperability between data processing tools, through shared schema and data type mechanisms (Oussous *et al.*, 2018). Coordination and workflow management consists of tools which coordinate applications and clusters in Big Data environments, simplifying distributed programming and storage (Oussous *et al.*, 2018) . Deployment management refers to systems which help while deploying the other layers of the Big Data stack by helping with authorization, cluster creation, easing the access to relevant libraries (Oussous *et al.*, 2018) .

HCatalog is a tool which is part of the Hadoop enterprise and its job is to manage table and storage metadata. As Figure 6 shows, HCatalog stands between the storage system and processing systems (HDFS, MapReduce, YARN), and the querying and analysis tools (Hive, Pig, Mahout). The objective is that HCatalog services data tables created in Hadoop, integrating with other services, using a common data model (Khan *et al.*, 2014). It simplifies user and service communication, providing shared schema and data type mechanisms. HCatalog also provides a relational view of the data in HDFS through its abstracted data table (Oussous *et al.*, 2018) .

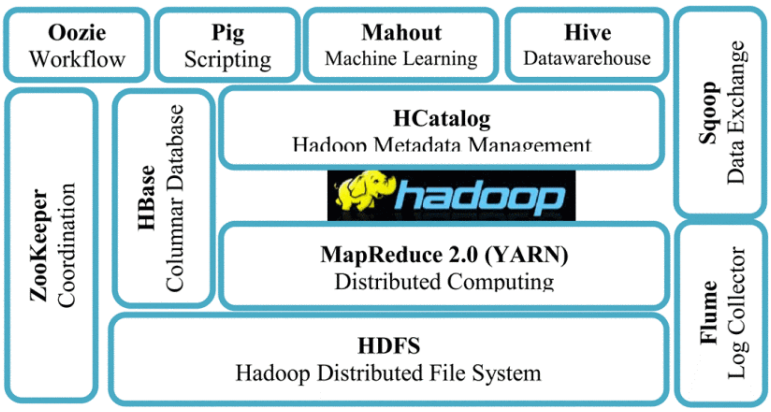


Figure 6:Hadoop Ecosystem (Alguliyev and Imamverdiyev, 2014)

Zookeper and Oozie are also part of the Hadoop environment as described in Figure 6. This service is designed to coordinate applications and clusters in the Hadoop environment,

providing high performance and data availability. Zookeper runs as a distributed application with a client-server architecture (Oussous *et al.*, 2018). Clients submit requests to Zookeeper through an API and presents abstractions to the clients of a set of data nodes for better organization of client metadata (Hunt *et al.*, 2019). Oozie meanwhile is a workflow scheduler which runs and manages jobs in Hadoop clusters (Oussous *et al.*, 2018). The workflows in Oozie take the form of Directed Acyclic Graphs (DAG) which map a sequence of actions and control dependencies (Bandi and Anitha, 2018). Alternatives outside Hadoop to Oozie include frameworks like Airflow (Singh and Singh, 2019).

To help with Hadoop deployment, tools like Ambari (Wadkar *et al.*, 2014), Whirr (Sammer, 2012), BigTop (Lovalekar, 2014) and Hue (Jenkins, 2016) exist. Ambari simplifies Hadoop management, supporting managing, provisioning, and monitoring through its web interface. Ambari also provides security to Hadoop by using Kerberos authentication (Oussous *et al.*, 2018). Whirr simplifies the creation of Hadoop cloud environments, and is used to deploy, configure, and start instances. BigTop provides integrity and reliability checks. Hue is another web application which allows interaction with the Hadoop ecosystem. Hue allows thew use of Hadoop without relying only on a command line, helping to browse the system, as well as creating jobs. Hue also provides a frontend for querying using Hive through Beeswax, which has wizards to help with queries, data load and also allowing for table creation, with integration to other tools like Impala (Oussous *et al.*, 2018).

2.2 Deep Learning

ML involves building mathematical models which aid with data analysis. Where “Learning” enters is when the insights extracted from data are parametrized into these models, which allows them to reflect how the data on which they were trained is characterized. When a model has been trained and tested, it can be used to make predictions, classifications, and understand new data observations (Cleary, 2019).

Traditional ML approaches fail to meet the standards required to analyse datasets with the volume, velocity, and variety of Big Data. Some approaches cannot produce results for this kind of data, nullifying the potential value of Big Data. Several ML algorithms are built with the presumption that an entire dataset can fit into memory. By using data representations instead of explicit features, Deep Learning tackles this issue (L’Heureux *et al.*, 2017).

In 2018 the European Commission launched a research and innovation programme, the European Union’s Horizon 2020 which leaned into various themes including "Excellent Science", "Industrial Leadership" and "Societal Challenges". This programme funded in part research in Deep Learning (Maia *et al.*, 2018) as it was an emerging technology. Deep Learning represents the world as a nested hierarchy of concepts where each concept is defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones in the form of neural networks. Due to its flexibility, Deep Learning has proven

to be a way to deal with the shortcomings of traditional ML (Zhou *et al.*, 2017). As shown in Figure 7, which details how different parts of an AI system relate to each other in different AI disciplines, Deep Learning adds one additional level to learn (grey boxes mean the component can learn from data) more abstract features of data (Lecun, Bengio and Hinton, 2015) .

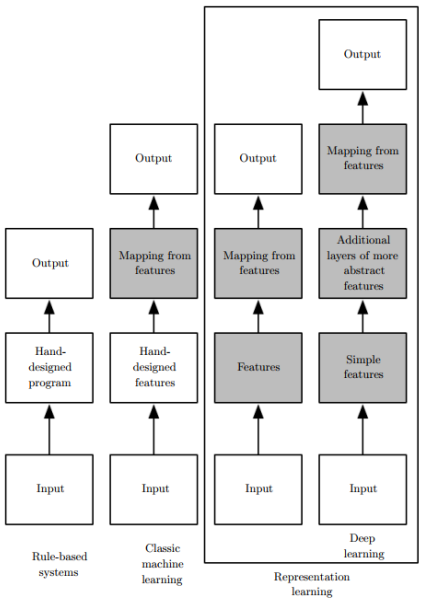


Figure 7: Comparison of different AI disciplines (Lecun, Bengio and Hinton, 2015)

This technology has gained track due to the availability of more computational power, from faster CPUs to the improvement of GPUs, faster internet connectivity and better software foundations (Zhou *et al.*, 2017).

Deep Neural Networks (DNN) are more complex versions of Artificial Neural Networks (ANN). ANN is an umbrella term and gets its name as an inspiration on biology by being designed to simulate the way the brain processes information (Agatonovic-Kustrin and Beresford, 2000). DNNs are composed of multi-layer interconnected nodes with more than one level of hidden layer nodes. Having multiple hidden layers allows for parameter learning and classification, all in the same network. DNNs gain the name due to number of hidden layers they possess as Figure 8 illustrates. DNNs can be trained in a supervised training mode, where the target attribute to predict or classify is available in the training data, or unsupervised mode, where training data is automatically generated from unlabelled data without much human intervention (Zhou *et al.*, 2017).

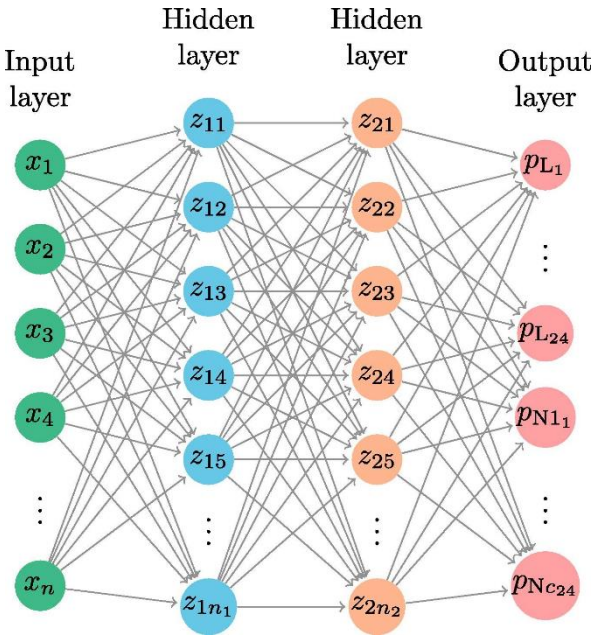


Figure 8: Illustration of a Deep Neural Network (Lago, De Ridder and De Schutter, 2018)

The Layers in a Neural Network contain an activation function which can be exchanged depending on the purpose of the network. Activations include the rectified linear units (ReLU), the sigmoid function, hyperbolic tangent (TanH), Softmax, among others as in Figure 9. The gradients of the ReLU activation function at the positive values becomes constant and do not vanish anymore. This means that the vanishing gradient problem can be avoided by using ReLU activation function.(Ide and Kurita, 2017). The perceptron, logistic sigmoid (or simply, sigmoid), and hyperbolic tangent are all activation functions that choose between two options (0 or 1, or in the case of tanh, -1 or1) (Johnson, Vulimiri and Zhang, 2020). In the case of a multinomial classification problem, a simpler network may be possible by using one-hot encoding. A one-hot encoding vector is defined for N exclusive options: one element in the N-element vector is 1, all other values are 0. Rather than using multiple layers to construct the binomial ladder required to simulate a multinomial decision, the Softmax activation function selects one-from-many in a single layer. (Johnson, Vulimiri and Zhang, 2020)

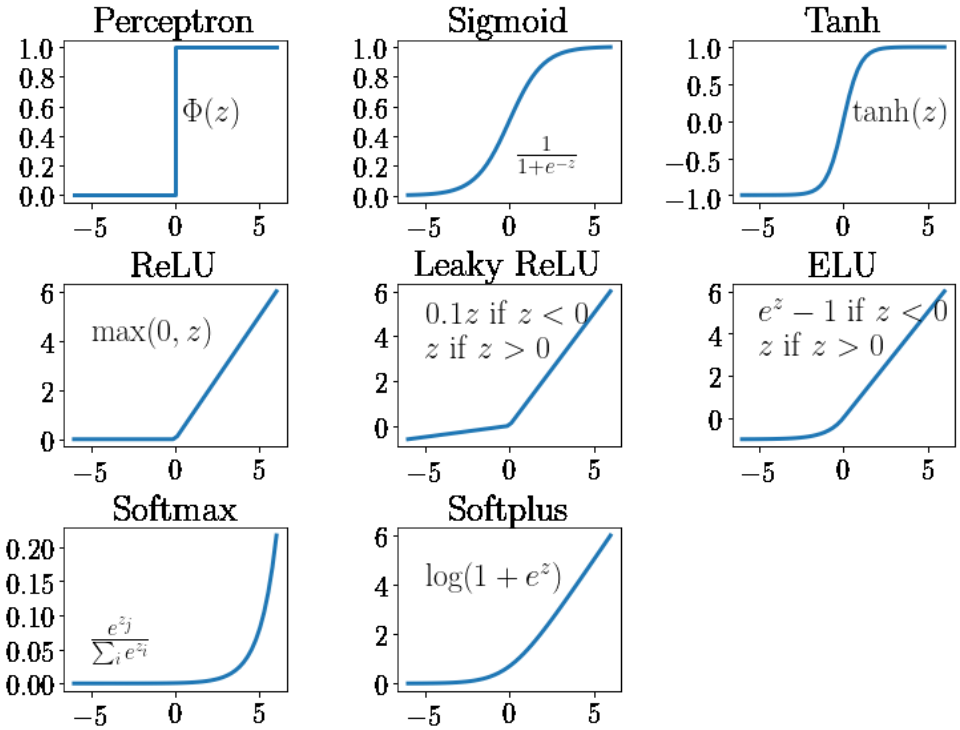


Figure 9: Different activation functions (Johnson, Vulimiri and Zhang, 2020)

Deep Learning has seen a large use in image and video processing, for image recognition systems, as it is able to break these complex data types into series of nested simple mappings, each described by a single layer of the model (Lecun, Bengio and Hinton, 2015). Convolutional Neural Networks (CNN) are mostly used for image and video processing and are a subtype of DNNs, which has been inspired by the visual cortex present in animals, that breaks inputs in smaller parts (e.g.: an image with 32 pixels being analysed 5 pixels at a time) as a way to keep the network at a manageable node size (Albawi, Mohammed and Al-Zawi, 2018). Table 5 shows some DNNs and their main uses.

Table 5 : Comparison of NN usage

	Image Processing	**NLP**	**Dimensionality Reduction**
Auto Encoder			✓
Convolutional NNs	✓	✓	✓
Long Short Term Memory		✓	✓
Gated Recurrent Units		✓	✓
Recursive NNs		✓	✓

Some Deep Neural Networks are capable of good results is NLP, by deriving vector representations of words based on how well they can predict the surrounding words in their context (Zhou *et al.*, 2017) . DNNs like the Auto Encoder, which is an unsupervised learning algorithm used for efficient dimensionality reduction, are ANNs with more hidden layers, which are used to perform feature learning before moving to classification, all in the same Network (Liu *et al.*, 2017). Recursive Neural Networks (RvNN) are also similar to ANNs and DNNs, depending on the depth, and what makes them recursive is the fact that its architecture allows the same set of weights to be applied recursively within a structural setting. RvNNs have a specific skewed tree structure as in Figure 10, which allows RvNNs to perform well in NLP (Irsoy and Cardie, 2014).

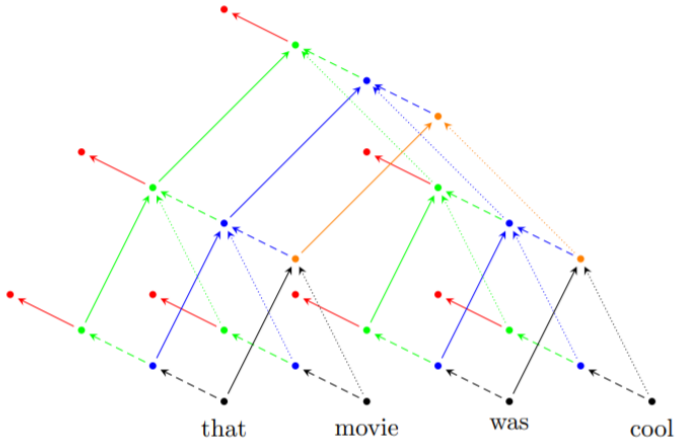


Figure 10: Three-layer deep RvNN (Irsoy and Cardie, 2014)

2.2.1 Deep Learning for Language Processing

NLP, also known as computational linguistics, is based on the use of computational models to solve problems in understanding natural languages. This area is divided in core and applications. Core NLP address the fundamentals of language modelling, responsible for quantifying associations among naturally occurring words, process syntax, parsing, morphological process, deal with sentence segmentation as well as identify parts of speech (PoS). The core of NLP builds sentence diagrams and provides semantic processing, which tries to derive the meaning of words and phrases. Applications on the other hand are broader in utility, and include translation of text, summarization, automatic question answering, classification and clustering of documents, and information extraction. Applications may need to handle core tasks in order to perform its function (Otter, Medina and Kalita, 2018) .

Attention based mechanisms have been used with DNNs for sentiment analysis and NLP. The attention-based mechanism allows the NN to focus on certain aspects of the input, allowing noise to be filtered out. The mechanism is usually implemented by assigning a real value score to each word or phrase, depending on the level of attention hierarchy implemented (Zhou, Wan and Xiao, 2016) .

CNNs (Figure 11) are hierarchical architectures and are good for extracting position invariant features (Yin *et al.*, 2017). CNNs are not largely associated to NLP by themselves, but when used with positional encoders they have proved to be useful for NLP classification. CNNs use convolutional, pooling and fully connected layers to process data (Rezaeinia *et al.*, 2019) .

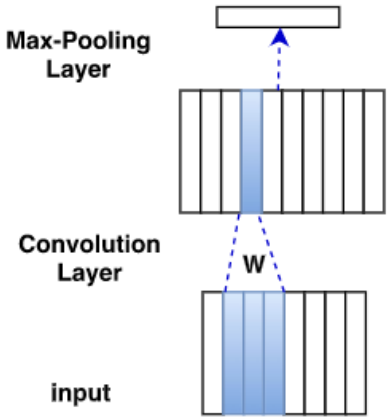


Figure 11: CNN structure (Yin et al., 2017)

RvNNs share weights vertically (between layers) to minimize training, which allows for modelling parse trees as the one in Figure 10. A single tensor (a generalized matrix) of weights can be used at a low level and recursively used at higher levels (Otter, Medina and Kalita, 2018).

Recurrent Neural Networks (RNN) are a subtype of RvNNs with a specific structure. NLP is dependent on the order of words or sentences, so it is useful to have a "memory" of previous elements when processing new ones due to backward dependencies (e.g.: semantic word meaning may depend on the words before or after it). RNNs accomplish this by combining the outputs of two layers, allowing phrase analysis in forward and backward directions, also called a bidirectional Recurrent NN (Otter, Medina and Kalita, 2018). Figure 12 represents an RNN with a single hidden layer, with the recursive notation in the left side, where X and Z are the input and output, respectively. The extended notation on the right shows the different steps in time (Chen, 2016) .

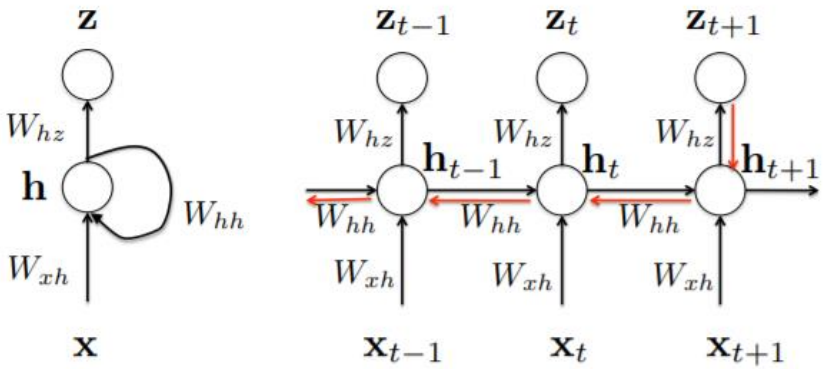


Figure 12: RNN example (Chen, 2016)

One other mechanism which may be used to improve on RNNs ability for NLP is to use Long Short-Term Memory (LSTM), which exchanges single recursive nodes for several individual neurons (interconnected nodes), as shown in Figure 13, connected in a manner designed to retain, forget or expose important information (Irsoy and Cardie, 2014). The structure in Figure 13 shows how the different logical gates interact with the Cell (Memory cell) which encodes the information of the inputs observed up to that step. The memory cell receives the same inputs (xt and ht-1, the output of the previous time step) and produces the same outputs (ht) as normal RNN would, the only difference being the gating units which control the information flow (Chen, 2016). The input gates (it and gt) control information going into the cell and output gate (ot) controls the information going away from it. The forget gate (ft) can reset the memory unit.

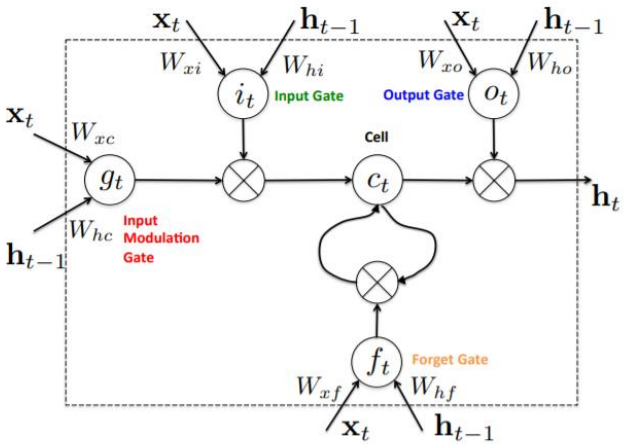


Figure 13: Unit structure of an LSTM(Chen, 2016)

In Oncharoen and Vateekul, 2018, an LSTM is implemented parallel to a CNN for Stock Market Prediction. The goal was to present a new model for Natural Language Processing, by including numbers in Language data analysis. The method which the work proposed was to use event embeddings (Ding *et al.*, 2015) which are obtained from feeding word embeddings into a neural tensor network which extracts the actor, action and object of an average of word embeddings. Event embedding allows for events to be interrelated as opposed from the word embedding method which only shows relations between words. This process can be used with different timeframes, returning the most prominent events in that timeframe. The event embeddings are then fed to different CNNs for long-term, mid-term and short-term pooling whose results are then fed to a SoftMax layer before being concatenated with the results from the LSTM which analyses stock data indicators (Open/High/Close/Low values of stock indexes) and feeds the results to a SoftMax layer.

2.2.2 Deep Learning Frameworks

Due to its rise in popularity, many frameworks have been developed to handle Neural Networks, more specifically Deep Neural Networks, which have been adopted for Deep Learning tasks. Deep Learning frameworks tend to have about the same features, with the main requirement being the training of NNs, most of them have support for CNNs, RvNN and DNNs.

Represented in Table 6, is an overview of some of the most popular open-source Deep Learning frameworks as well as some of their common features. Graph Type refers to the way a graph is generated. While a static frameworks generate a graph structure which is differentiated symbolically ahead of time and then run many times, a dynamic one defines the function to be differentiated simply by running the desired computation (Paszke *et al.*, 2017a). Chainer and PyTorch are examples of frameworks which dynamically generate graphs (Tokui *et al.*, 2015; Paszke *et al.*, 2017b)

Table 6: Overview of Deep Learning Frameworks

Framework	Graph type	Back Propagation	Forward Propagation	Parameters	Possible NNs	User language
Theano	Static	✓	✓	Separate Nodes	DNN; CNN; RvNN	Python, C++, CUDA
Caffe	Static	✓	✘	4-dimensional arrays called blobs	DNN; CNN; RvNN[2]	Python, C++, MATLAB
Tensorflow	Static[3]	✓	✓	Separate Nodes	DNN; CNN; RvNN	Python
Chainer	Dynamic	✓	✘	Separate Nodes	DNN; CNN; RvNN	Python
PyTorch	Dynamic	✓	✘	Separate Nodes	DNN; RvNN	Python

Another feature which is prominent in Deep Learning frameworks is backpropagation, which allows the computation of gradients on particular nodes with respect to one or many other nodes. Backpropagation uses the gradient to balance of the weights associated to the nodes by applying a backward propagation chain rule through the graph, adjusting the weights in the output to input direction (The Theano Development Team *et al.*, 2016). All the frameworks presented in Table 6 have this feature. Where this varies however is in the presence of forward propagation, where Theano and Tensorflow transverse the NN in the input to output direction to calculate the cost as opposed to the gradient (Goldsborough, 2016; The Theano Development Team *et al.*, 2016) .

The most common use case for the NNs generated from Deep Learning is for data to be stored in a network of layers. Each layer is composed of vertexes (nodes), and edges which connect each layer's node to the ones in the subsequent layers, where each node indicates a variable, which can be a scalar, vector, matrix, tensor or even a type of another variable (Bengio, 2009)., Caffe approaches this in a different way, by implementing a concept which

[2] RvNN support has been added recently to Caffe (Bahrampour *et al.*, 2015)

[3] Tensorflow can support dynamic computations using dynamic control flow

has been named *blob,* a 4-dimensional array. Caffe can hold batches of data between each layer. This allows the framework to ignore low level details while maintaining a high level of performance (Jia *et al.*, 2014).

The frameworks in Table 6 are available in different programming languages and may be used to build integral system components. There is also the possibility of building these Deep Learning systems as microservices by using a Keras, a Deep Learning API with a switchable backend, compatible with other frameworks (Afonso *et al.*, 2018). Keras is designed to reduce the cognitive load of its users, shifting the focus from the implementation details and allowing to focus on the creation of models (Grattarola and Alippi, 2021).

Another library, which is not included in Table 6, is fast.ai, which has the same functionalities as PyTorch, in addition to ease of use features included by the library which is itself an abstraction layer built on top of PyTorch (Doleck *et al.*, 2020).

2.3 Social Networking Data

Social Networks connect individuals or groups of people with similar interests and features. In more recent years, these networks have started connecting those same individuals to software components, such as Web-based services, data resources and workflows. All these interactions between the different components in ever-expanding networks provide continuous streams of data which can be used to gather insights on crowd wisdom and reveal various patterns. In a real-world example, the 2013 Boston marathon bombings' perpetrators were found by collecting, combining, clustering and analysing different data objects, like videos and images of different views and angles posted by a number of social network users (Tan *et al.*, 2013).

Understanding social networks has evolved to become one of Big Data's problems, where many specialists hope to understand and predict human behaviour to empower marketing, sales, and online commerce (Tan *et al.*, 2013) . As of 2018 Twitter had 310 million subscribers, and 230 million tweets per day, YouTube had more than one billion users globally having a presence in 88 countries with 76 different languages and a billion hour per day of video playback, Facebook created 30 petabytes of data per day (Can and Alatas, 2017).

2.3.1 Social Network Data Studies

In 2012, attendees of the Davos World Economic Forum declared Big Data to be a strategic economic resource, as important as money and gold. In 2018 the UN Secretary General tasked the Independent Expert Advisory Group to prepare a report, which defined the data revolution for sustainable development as producing better quality data to monitor sustainable development by combining data from new technologies with traditional data. This means that new technologies like social networks were included, which effectively allowed for

policy and decision-making to be affected by the billions of users and their data worldwide (Can and Alatas, 2017) .

Big Data has been employed to make insights of social network data, and from this, many new methods were adopted for its analysis, like anomaly detection (Prasad, Almanza-Garcia and Lu, 2009), discrimination discovery (Ruggieri, Pedreschi and Turini, 2010), opinion leaders detection (Cho, wang and Lee, 2012), event detection (Zhou and Chen, 2014), role mining (Rossi and Ahmed, 2015), rumour propagation detection system (Zhang *et al.*, 2015), conflict detection (Bonchi and Ferrari, 2010), and topic detection (Cataldi, Di Caro and Schifanella, 2010). Social network data also helped contribute to women's rights by collecting communications made by women and creating an image of their social and economic situation, education, working conditions, and harassment in order to help strengthen the position of women in society, as the International Labour Organization did in the Asia-Pacific region (Can and Alatas, 2017).

The environment has also benefited from social network data analysis, by allowing sustainable policies to be developed and monitored based on people's reactions to them. During the 2014 Climate Summit, the Secretary General's Climate Change Support Team created a tool, with the support of Global Pulse, to monitor social media interactions on climate-change related issues in real-time. The insights extracted from this tool were used in the development of environmental policies (Can and Alatas, 2017) . In Nigeria, studies of social media data were called to shape multinational companies' accountability in the environmental protection of the Niger Delta as a way to push for a more sustainable future (Nwagbara, 2013).

In the Healthcare department, social network platforms allow health platform access without economic or social distinctions. Social networks have also been used as a medium in which people can share their healthcare experiences and help raise awareness for health issues, aiding in the prevention of diseases. Google used social data to monitor the spread of the flu and claimed people could be warned of flu outbreaks as far as two weeks prior (Can and Alatas, 2017) . In 2020, studies were developed on how to help fight the COVID-19, on topics such as contact tracing, herd immunity as well as simulations of policies like lockdowns and social distancing based on social data (Karaivanov, 2020).

Another field where social data has been shaping is the financial sector. Today companies can have their image and therefore their investment influenced by social media. Platforms like Twitter, Instagram and Facebook have become a forum for communication between people and businesses and at the same time a channel which allows the same business to market themselves in an unexpensive way, contributing to company branding (Can and Alatas, 2017) . Recently the social network Reddit was the centre stage for financial market turmoil which caused hedge funds to lose millions of dollars due to an in mass acquisition of stock in companies like GameStop and AMC (Muzio, 2021).

2.3.2 Machine Learning on Social Network Data

Social Networks provide unstructured data, in the form of comments, posts, tweets, etc. Which, in their standard form, are not fit for analysis, requiring pre-processing. Two ways this pre-processing can be done is either via NLP which extracts semantics associated to symbols in documents, and sentiment analysis, a clustering algorithm which identifies groups of user entries with similar properties (Khanaferov, Luc and Wang, 2014) .

In Prabha and Srikanth (2019), several works were analysed to understand how sentiment analysis is performed in the academic world. In Hassan and Mahmood, 2018 an hybrid approach of CNN and RNN was used for sentence classification. The RNN was introduced to counter the drawbacks on CNNs pooling layers, which focus on high level features and ignore others. RNN however suffers from vanishing and exploding gradient, which motivated the adoption of LSTM to overcome this issue. When compared to traditional methods like SVM, Naïve Bayes and other CNN forms the model outperformed all others in terms of accuracy, proving that pooling layers degrades performance and RNN-LSTM can achieve the same results with a smaller architecture.

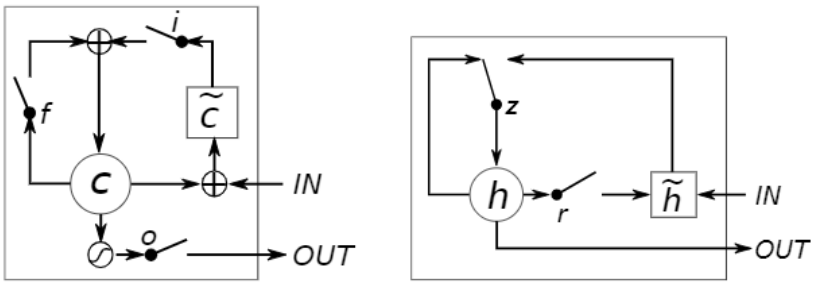
In Nguyen and Nguyen (2018), a deep CNN and a bidirectional LSTM are proposed for Twitter social data analysis. Deep CNN was applied on character-level embeddings to increase word embedding information. This work includes a tweet processor which removes non-essential words from tweets while retaining necessary information and applies a set of semantic rules. The output of the tweet processor is fed to the Deep CNN which is trained on the character embeddings. The output of Deep CNN as well as word embeddings from pre-trained word vectors are concatenated and fed to the bidirectional LSTM. When tested on Twitter datasets and compared to traditional ML approaches, this method outperformed them. Separate Deep CNN and bidirectional LSTM were also trained and tested to prove that the combination of both produces better results (Prabha and Srikanth, 2019).

In Basiri *et al.* (2021), there is a comparison of models using different datasets, three of which are from Twitter. Sentiment Specific Word Embedding (SS-BED), hierarchical attention networks (HAN), improved word vectors (IWV), a recurrent convolutional neural network with attention for sentiment Classification (ARC) and a bidirectional LSTM with attention mechanism and convolutional layer (AC-BiLSTM).

SS-BED is proposed in Chatterjee *et al.* (2019) as an approach which leverages both the sentiment and semantic representations of user utterance for accurate emotion detection. SS-BED proposes two parallel LSTM layers on two different word embedding matrices which are then fed to a fully connected network with one hidden layer to predict emotion categories.

In Yang *et al.* (2016), HAN is proposed, which takes advantage of gated recurrent unit (GRU), another subtype of RNN and a variant of LSTM (Figure 14) which uses a gated mechanism to track the state of sequences without using separate memory cells. The model consists of a word sequence encoder, a bidirectional GRU, a word-level attention layer used to form a weighted sentence vector, a sentence encoder, another bidirectional GRU, and a

sentence-level attention layer which rewards sentences which correctly classify a document. Figure 14 illustrates the difference between GRU and LSTM. *i*, *f* and o are the input, forget and output gates, respectively. *c* and ~*c* denote the memory cell and the new memory cell content. *r* and *z* are the reset and update gates, *h* and ~*h* are the activation and the candidate activation. The mechanism is very similar, however, the memory cell is not separate. The difference in behaviour is that unlike the LSTM, a GRU cell exposes its whole state to the other cells in the networks, while the LSTM controls the exposure in the output gate.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 14: GRU and LSTM configuration (Chung *et al.*, 2014)

Rezaeinia *et al.* (2019) introduced the IWV, which is based in the Part-Of-Speech (POS) tagging techniques (Schmid, 1994), lexicon-based approaches, word position algorithm, and word2vec and GloVe (Shi and Liu, 2014) methods. The employed CNN consists of three convolution layers, a max pooling layer and a fully connected layer designed for sentiment polarity classification.

Wen and Li (2018) explores a combination of RNN and CNN for sentiment classification as well as an attention mechanism to effectively capture important emotional content. The RNN used in this context is a GRU, more specifically a one-layer bidirectional GRU, applied to word vectors and fed to an attention layer. The output of the attention mechanism is then fed to a CNN layer, followed by max-over time pooling operations and a fully connected layer.

The paper Liu and Guo (2019) studies the application of bidirectional LSTM networks for natural language processing and sentiment analysis. A one-dimension CNN layer precedes the LSTM network and uses different filter sizes for local feature extraction. After the CNN layer and the LSTM layer, the results are fed into an attention mechanism. The output is a dropout layer, a regularization technique to avoid overfitting, and a SoftMax layer, a multi-category classification technique, which generates conditional probabilities over the class space to achieve classification.

2.3.3 Social Network Data for Financial Prediction

In Sohangir *et al.* (2018), Financial Social Network data is used to predict the movement of the stock market by employing Deep Learning. By using data from the StockTwits social network

which consists of up to 140-character market-oriented posts and specific labels on data sentiment. This paper used the Pearson Correlation Coefficient (Pearson, 1895) to validate if stock prices follow a linear relation to user sentiment, which returned a 53% correlation on general authors, which increased to 75% when predictions were filtered by the highest ranking authors' predictions. Initially, Naïve Bayes and Support Vector Machines (SVM) and Decision Trees were used for NLP with SVMs producing the highest accuracy (76.2%). After introducing feature selection to reduce noisy data within the dataset coupled with regression, ANOVA produced a consistently higher accuracy than other methods (70%) with a lower level of features. Moving to Deep Learning, Doc2Vec, a generalized version of word2vec which produces high dimensional vector representation of documents using Deep Learning, was used with a maximum of 67% accuracy. LSTM was then used with 69% accuracy and finally, CNNs with up to 70000 convolutional steps, which provided the highest accuracy (98.9%) and outperformed regression after less than 2000 steps (79% accuracy), showing the potential of Deep Learning for this specific area of prediction of financial markets based on social network data.

This section describes how Big Data has been explored. Technologies to store, process, query, access, analyse and manage Big Data are described. This section also provides a background on Deep Learning, by explaining what it is, how it is done and common techniques. The focus is on Deep Learning for Sentiment analysis, so there is also an explanation of some language processing techniques, specifically those which were applied in Social Network related projects.

The next section explains the value this work will bring to the research on the topic of Big Data analytics by using Deep Learning to deal with the theme of social networks for financial prediction as an example.

3 Value Analysis

Value Analysis' main goal is to assess how to increase the value of a product or service at the lowest cost to quality ratio. It is an organized creative approach which allows for identifying unnecessary cost, which is cost that does not provide any quality, use, appearance, or costumer features. Value analysis is:

1. A systematic, formal, and organized process of analysis and evaluation.
2. A management activity which requires planning, control, and coordination.

Analysis is based upon the function of a product or a service to meet the demands of its users, which must include an understanding of purpose of the thing in analysis at a review process. To understand the use of a product/service, it must be accepted that specifications to assess the level of fit between the product/service and the value derived by users can be established. To deliver value to the user, the functional specification must be met. The formal review process should end in a process of design improvements which allow production costs to go down while maintaining the quality (Petrovic and Kittl, 2003).

3.1 Opportunity Identification

Big Data, characterized by its 5 V's (Value, Volume, Velocity, Variety, Veracity), which translate into features to take advantage, but come coupled with problems (Gudivada *et al.*, 2015):

- **Value** – Analysis of Big Data can provide counterintuitive insights. Concept Drift is one issue which may be caused by changes in the conditional distribution of data output while maintaining the same input.
- **Volume** – Big Data is already in the terabytes (2^{40}) and petabytes (20^{50}) and rapidly heading into exabytes (20^{60}).
- **Velocity** – Data is produces in very high rates and may requires real-time processing to determine whether to store this data. Velocity requires incremental learning to adapt its learning to the arrival of new data.
- **Variety** – Data may exist in all forms, some structured or semi structured or even completely unstructured. Dirty and noisy data is especially difficult to find in such a variety of information.
- **Veracity** – Due to the diversity among data sources and intermediary processing, concerns are raised about security, privacy, trust, and accountability. Data provenance is a must.

Besides these challenges, Big Data is still a rich source for information which is seen as a revolution which will transform how we live, so there is an effort to extract the knowledge within Big Data to enable better decision making (L'Heureux *et al.*, 2017).

Proposed solutions to the problems of Big Data include data manipulations, like dimensionality reduction or instance selection, processing manipulations like scaling (vertical and horizontal), and algorithm manipulations by implementing new paradigms. These solutions may be used in tandem. Deep Learning is a type of paradigm change in ML algorithms, which uses DNNs to extract features from data prior to the machine learning tasks and may be useful to address some of the problems of Big Data (L'Heureux *et al.*, 2017).

3.2 Orientation

The Quality Function Deployment (QFD) is a technique designed to ensure that customer needs are addressed in new project. This method starts by gathering requirements from customers and translating them into engineering characteristics. In QFD, quality means meeting customer needs, function is the focus on what must be done, and deployment is for who will do it and when. One of the outputs of QFD is the House of Quality (Figure 15) which allows the conversion of customer attributes into engineering characteristic (Group, 2013).

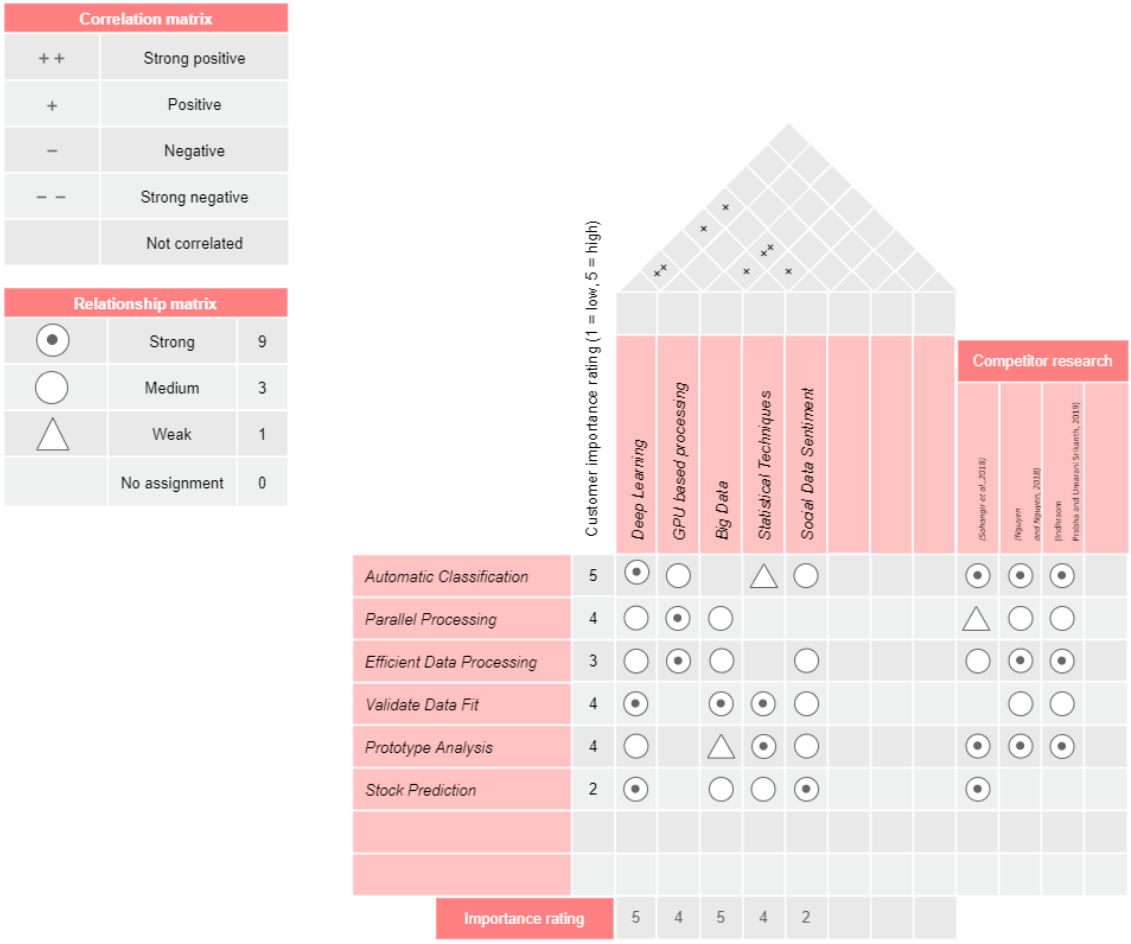


Figure 15:Project House of Quality

The main requirements provided for this project are as follows:

- Implement an automatic classification system based on Deep Learning.
- Validate data fit and resolution.
- Analyse the prototype's performance.
- Take advantage of technologies like GPU for parallel processing.
- Take advantage of technologies like GPU for processing power.
- Efficient processing for high data volumes.
- Use stock prediction as a use case based on social data sentiment.

In Figure 15 there is the House of Quality which translates the requirements above into engineering characteristics.

The competitors in Figure 15 are, from left to right, Sohangir *et al.* (2018), Nguyen and Nguyen (2018), and Prabha and Srikanth (2019).

3.3 Functional Identification and analysis

With the engineering characteristics defined in the precedent subsection it is possible to define the most important functions of the system. A function in this context is the demanded use of a system part and the esteem it produces. These functions are what makes the system work as expected. To define a basic function in a simple way, a verb and a noun are used in combination (Rains, 2009).

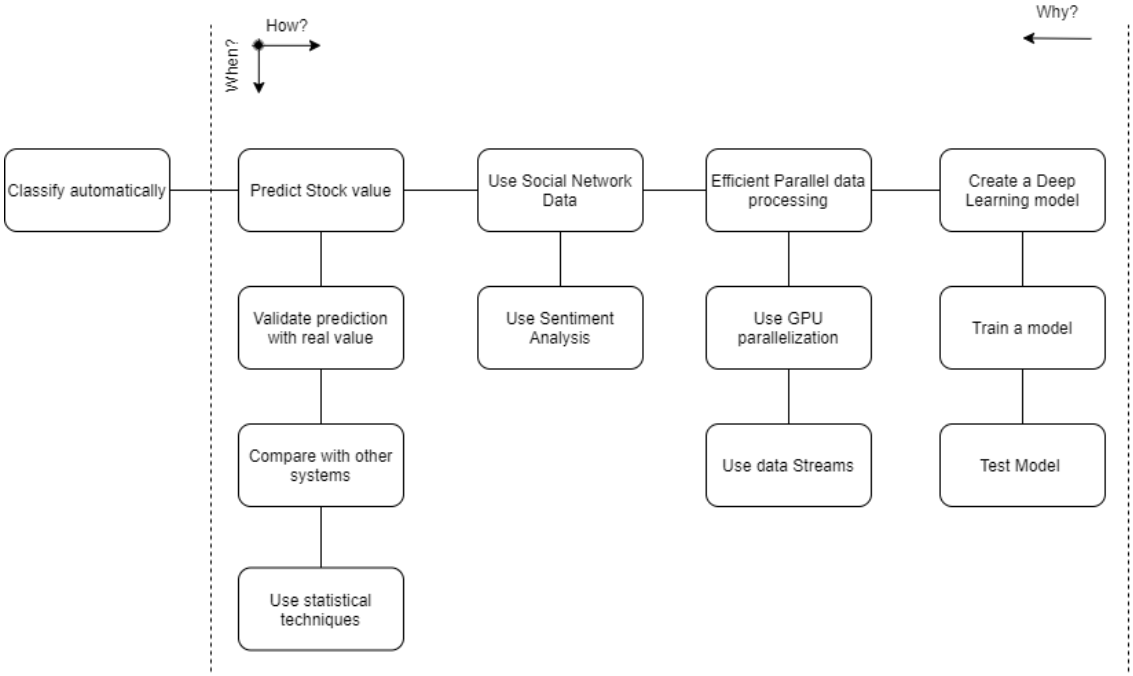


Figure 16: FAST analysis

The systematic function definition starts by answering 'What function does this system undertake?' which will produce the fundamental function, which serves as the starting point for further function selection. The Function Analysis System Technique (FAST) provides a way to group the functions in a structured and logical manner. FAST is a technique to develop a graphical representation of functional relations based on the questions "How" and "Why".

3.4 Creative Alternatives and Analysis

Given the functional identification in the previous subsection, the functions identified were those of automatic classification, using social network data, via sentiment analysis, using a Deep Learning model to predict sock value. Deep Learning however can be developed in many ways, and different approaches have been used for sentiment analysis of text data.

To decide on the Deep Learning approach to implement, the Analytic Hierarchy Process (AHP) has been adopted. AHP is a multi-criteria decision analysis method which allows the prioritization of alternatives in conflicting criteria scenarios, aiming to reach a compromise (Buchanan and Gardiner, 2003) . AHP works by dividing the problem in a hierarchy to ease comprehension and evaluation, and is composed of the following phases:

1. Build a hierarchical decision tree.
2. Compare criteria alternatives.
3. Relative criteria priority.
4. Evaluate the consistency of the relative priorities.
5. Build a pair comparison matrix for each criterion, considering each of the considered alternatives.
6. Obtain the composite priority for the alternatives.
7. Select the alternative.

The decision to be taken in this context is which paradigm to implement for the Deep Learning Network. From section 2.3, there are a few alternatives for Deep Learning implementation, which include CNNs, RvNNs and both subtypes, RNNs and LSTMs, GRUs, an LSTM subtype, combinations of RNNs with CNNs (Hassan and Mahmood, 2018) and also LSTM and CNN combinations (Nguyen and Nguyen, 2018). The hierarchic tree is the composed of:

- **Objective**: Deep Learning Paradigm
- **Criteria**: Accuracy, Precision, Recall, F1 Score
- **Alternatives**: CNN, LSTM, GRU, CNN+RNN, CNN+LSTM

The first phase of AHP is to build the hierarchical decision tree represented in Figure 17 with the elements mentioned previously.

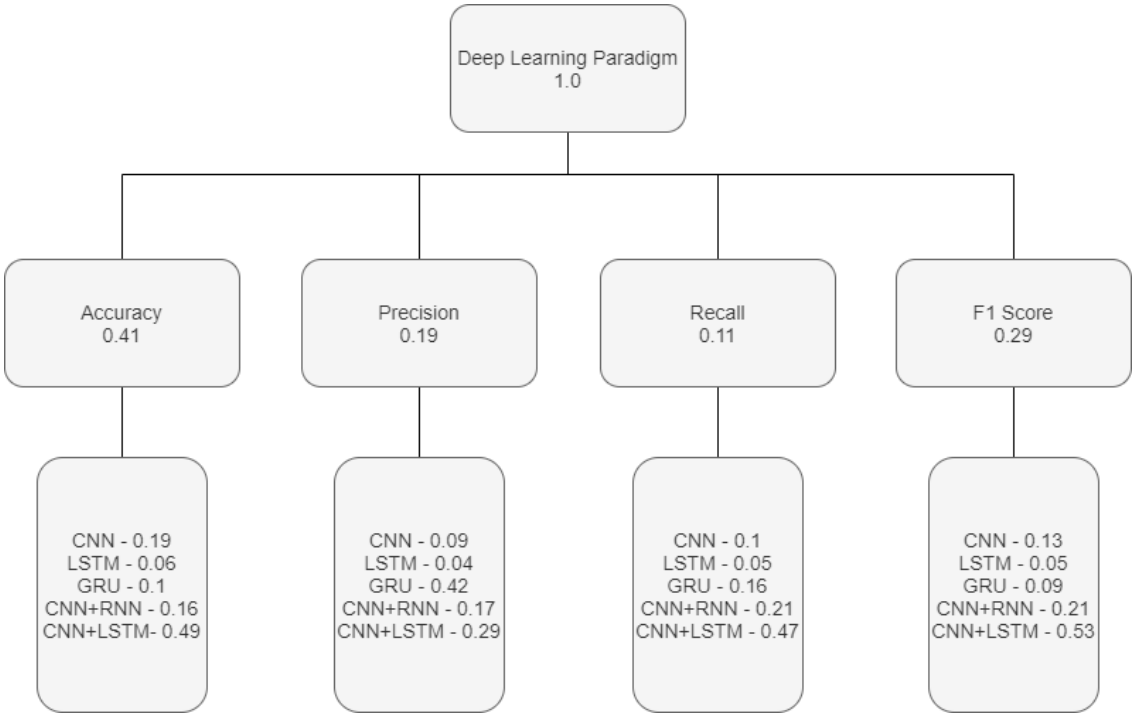


Figure 17: AHP hierarchical decision tree

The second phase of AHP is to build the comparison table for the different levels of the tree using the fundamental scale for importance level comparison (Saaty, 1987). Table 7 represents the comparison on the criteria, Table 8 has the accuracy comparison, Table 9 the precision comparison, Table 10 recall comparison, and Table 11 the F1 Score comparison. Values in the alternatives tables were based on the data from *ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis* (Basiri *et al.*, 2021) as explained in section 2.3.2.

Table 7: AHP criteria comparison

Criteria	Accuracy	Precision	Recall	F1
Accuracy	1	2	3	2
Precision	1/2	1	2	1/2
Recall	1/3	1/2	1	1/3
F1 Score	1/2	2	3	1

Table 8: Accuracy Comparison

Alternatives	CNN	LSTM	GRU	CNN+RNN	CNN+LSTM
CNN	1	3	2	2	1/4
LSTM	1/3	1	1/2	1/3	1/6
GRU	1/2	2	1	1/2	1/5
CNN+RNN	1/2	3	2	1	1/3
CNN+LSTM	4	6	5	3	1

Table 9: Precision Comparison

Alternatives	CNN	LSTM	GRU	CNN+RNN	CNN+LSTM
CNN	1	4	1/5	1/3	1/4
LSTM	1/4	1	1/8	1/6	1/7
GRU	5	8	1	3	2
CNN+RNN	3	6	1/3	1	1/3
CNN+LSTM	4	7	1/2	3	1

Table 10: Recall Comparison

Alternatives	CNN	LSTM	GRU	CNN+RNN	CNN+LSTM
CNN	1	3	1/2	1/2	1/5
LSTM	1/3	1	1/4	1/4	1/7
GRU	2	4	1	1/2	1/3
CNN+RNN	2	4	2	1	1/3
CNN+LSTM	5	7	3	3	1

Table 11: F1 Score Comparison

Alternatives	CNN	LSTM	GRU	CNN+RNN	CNN+LSTM
CNN	1	3	2	1/2	1/5
LSTM	1/3	1	1/2	1/4	1/7
GRU	1/2	2	1	1/3	1/5
CNN+RNN	2	4	3	1	1/4
CNN+LSTM	5	7	5	4	1

The next step is to use the criteria comparison table to obtain each criterion's relative priority by first normalizing Table 7, which is done by multiplying each value for the sum of its column, which is pictured in Table 12. The result is in Table 13, along with each line's relative priority, which is achieved by averaging each row.

Table 12: AHP criteria comparison row sum

Total	2.33	5.50	9.00	3.83	2.33

Table 13: AHP normalized criteria with relative priority

Criteria	Accuracy	Precision	Recall	F1	Relative Priority

Accuracy	0.43	0.36	0.33	0.52	0.41
Precision	0.21	0.18	0.22	0.13	0.19
Recall	0.14	0.09	0.11	0.09	0.11
F1 Score	0.21	0.36	0.33	0.26	0.29

According to Table 13, the most significant criteria is accuracy with 0.41 relative priority, followed by F1 score, precision, and the least significant, recall.

Next, to check if the values obtained in the criteria comparison are feasible, the relative priority consistency is evaluated using the consistency reason (RC). AHP is based in the rational deciding assumption, that is, if A is preferred to B, and B preferred to C, then A must be preferred to C. Using RC, in the case its value is higher than 0.1, judgement is not trustworthy as they are too close to randomness. To obtain the RC value, λ_{max} is needed, which can be obtained via the equation:

$$Ax = \lambda_{max}x,$$

where A is the is the comparison matrix in Table 7 and x the relative priority matrix in Table 13.

When λ_{max} is known, the consistency index (IC) can be obtained via the equation:

$$IC = \frac{\lambda_{max} - n}{n - 1}, \text{where n is the number of criteria}$$

RC is then obtained via the equation:

$$RC = \frac{IC}{IR}, where\ IR\ is\ an\ index\ of\ \text{Table 14}$$

Table 14: IR values for squared matrixes of order n

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

The IR values in Table 14 are a random index calculated by the National Oak Ridge Laboratory in the USA for squared matrixes of order n. Each number of the table is the result of averaging the ICs from randomly selected matrixes.

The λ_{max}, calculated as in the formula $Ax = \lambda_{max}x$, is 4.07, which gives an IC value of 0.023, and an RC of 0.026 (Appendix A(1)), which is lower than 0.1, meaning that the relative priorities are consistent.

To validate which alternative is the best option to take, relative priorities must be calculated for each of the criteria tables, by first normalizing the matrixes, dividing each element by the sum of its column, and then averaging each row (Appendix A(2)). The values in Figure 17's bottom level, are the result of these relative priority vectors. Which can then be

used to produce the composite priority of the alternatives by multiplying the alternatives relative priority matrix with the criteria relative priority vector.

$$\begin{bmatrix} 0.19 & 0.09 & 0.10 & 0.13 \\ 0.06 & 0.04 & 0.05 & 0.05 \\ 0.10 & 0.42 & 0.16 & 0.09 \\ 0.16 & 0.17 & 0.21 & 0.21 \\ 0.49 & 0.29 & 0.47 & 0.53 \end{bmatrix} \times \begin{bmatrix} 0.41 \\ 0.19 \\ 0.11 \\ 0.29 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.05 \\ 0.16 \\ 0.18 \\ 0.46 \end{bmatrix}$$

With the result of the matrix multiplication, knowing that the lines in the matrix are ordered the same way as they are in the alternative comparison tables (CNN, LSTM, GRU, CNN+RNN, CNN+LSTM in this order), the resulting vector exhibits the results in the same order, which results in CNN+LSTM being the most relevant alternative in function of the defined criteria and respective importance.

4 Methodology

The goal of this section is to explain how the work on the system was planned, as well as list and justify the choices which were made. Section 4.1 starts by presenting the selected Deep Learning framework for the system, complemented by rational and factual evidence for the choice. The strategy adopted for the work will be explained as well, in section 4.2. Section 4.2 contains a brief explanation of how work was organized. The planned architecture for the system, complete with alternatives is present in section 4.3. The datasets and data sources which will be used will also be listed in this section.

4.1 Deep Learning Framework Alternatives

In section 2.2.2, there is a list of Deep Learning Frameworks as well as a comparison table with some characteristics such as the possibility of backpropagation or forward propagation, the possible DNNs, etc. The scope of this dissertation is not to study which of the frameworks provides the best results, so the decision on which to use was made based on existing papers, and the popularity of the frameworks for solving the problem which this dissertation addresses.

In Doleck *et al.* (2020), a performance comparison of different frameworks is made, using two distinct datasets:

- The MOOC dataset, which includes 6241 instances of video viewing features tabulated on a weekly basis (number of videos viewed per week, number of stops, pauses, etc.);
- The CEGEP academic performance dataset which includes 309 instances of student information, such as age, gender, prior academic performance, etc.This paper analysed Keras (using a Tensorflow backend and a Theano backend), Tensorflow using Decision Trees for Gradient Boosting, Pytorch, and fast.ai, all in terms of accuracy.

From the tests using the MOOC dataset, the results displayed in Table 15 show that Keras with a Tensorflow backend has an overall best performance.

Table 15: MOOC framework accuracy results (Doleck *et al.*, 2020)

Framework	Accuracy
Keras (Using Tensorflow backend)	From 58.29% to 69.19%
Tensorflow (BoostedTrees)	63.13%
Keras (Using Theano backend)	From 66.37% to 69.00%
Pytorch	From 67.26% to 68.25%
Fast.ai (Using Pytorch backend)	68.19% with a standard deviation of 2.76%

The tests using the CEGEP dataset, the results from Table 16 show that Tensorflow (BoostedTrees) has a better overall performance.

Table 16: CEGEP framework accuracy results (Doleck *et al.*, 2020)

Framework	Accuracy
Keras (Using Tensorflow backend)	From 62.85% to 85.13%
Tensorflow (BoostedTrees)	90.32%
Keras (Using Theano backend)	From 62.20% to 84.82%
Pytorch	From 76.05% to 84.79%
Fast.ai (Using Pytorch backend)	88.55% with a standard deviation of 3.71%

Given that Chainer and Caffe are not represented in this comparison, another analysis, regarding the popularity of the different frameworks for Deep Learning sentiment analysis was made. By using the dimensions.ai app (Orduna-Malea and Lopez-Cozar, 2018) which indexes academic research papers (just as Google Scholar) and allows the analysis of various aspects related to them, it was possible to obtain a statistic of the usage of each framework. For this comparison, the query string to search in dimensions.ai used the string "*Deep Learning AND sentiment analysis*" as well as an "*AND*" for each of the frameworks.

Figure 18 shows the number of scientific papers which mention each framework, by year, extracted from the dimensions.ai website. The mentions of the selected frameworks show that Tensorflow has been the most widely used framework since 2015. The numbers decrease when the condition of not mentioning Keras is added to the search, but still maintains the lead with Caffe in second place.

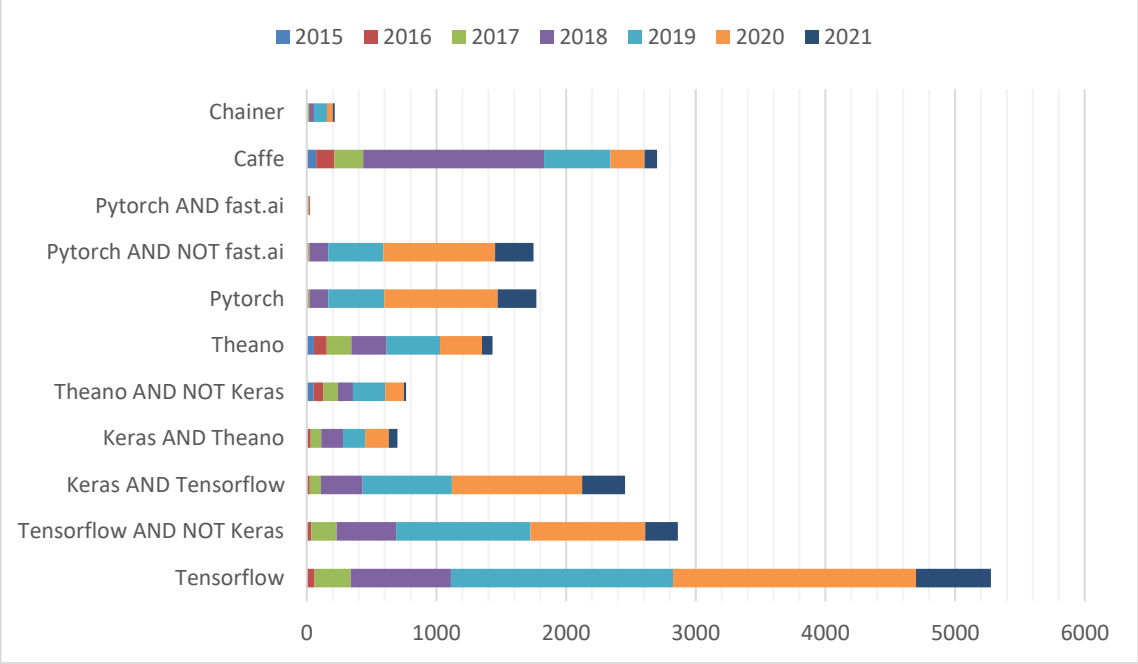


Figure 18: Aggregate mentions for Deep Learning Frameworks

What Figure 19 shows is that Caffe use peaked in 2018 and has been declining ever since as opposed to Tensorflow which has only been increasing. Keras usage with Tensorflow has been increasing and has surpassed the use of standalone Tensorflow in 2020.

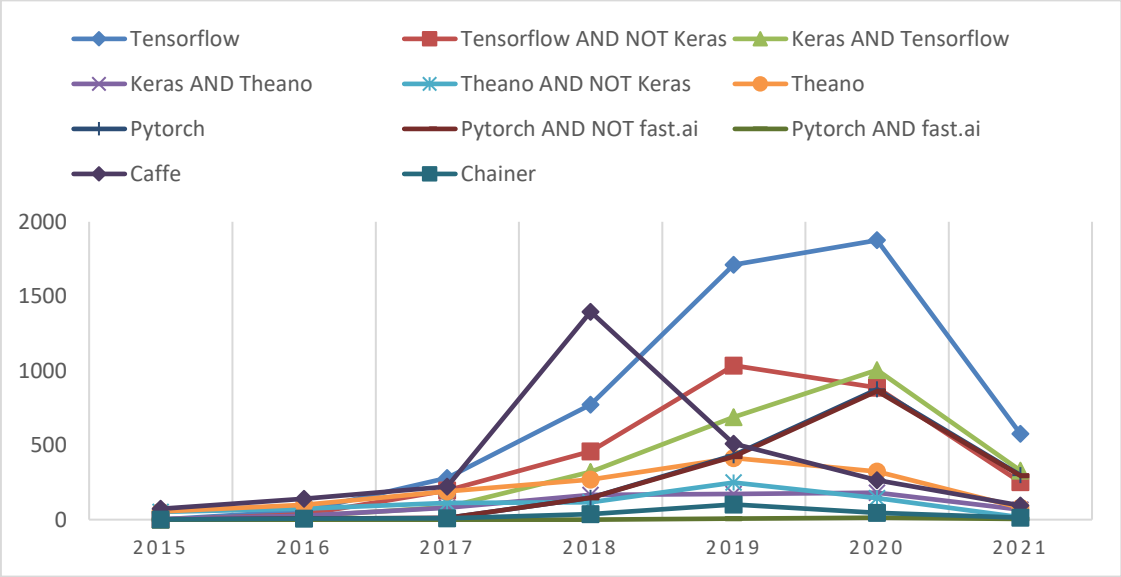


Figure 19: Mentions for Deep Learning Frameworks by year

Having analysed all the aspects above, we can see that a trend is rising on the usage of Keras and Tensorflow. Tensorflow on its own has been used the most when comparing the database of scientific research papers, and since it has been developed, Keras has started gaining traction fast, being mostly coupled with Tensorflow. Additionally, when coupled with Keras, Tensorflow registered a more balanced set of results on the accuracy tests. Having these considerations in mind, Keras with a Tensorflow backend was selected to be used in this project.

4.2 Work Process

This work implemented the Cross Industry Process for Data Mining (CRISP-DM) methodology (Wirth, 2000) illustrated in Figure 20.

In the CRISP-DM methodology, the phases are as follows:

- Business Understanding: Focusing on understanding the project's objective and requirements from a business perspective, and then converting this knowledge into a data mining problem definition, and preliminary project plan.
- Data Understanding: Initial data collection and activities to get familiar with it, identify problems, discover the first insights, and detect interesting datasets within the whole data. The formulation of the data problem requires some understanding of the available data.

- Data Preparation: Activities to construct the dataset to feed into modelling tools from the initial raw data. Data preparation tasks are likely to be performed multiple times and in no specific order. Tasks include table, record, and attribute selection, data cleaning, construction of new attributes, and data transformation for modelling tools.
- Modelling: Various techniques are selected and applied, with their parameters calibrated to their optimal values. There are various modelling techniques for the same data problem. This phase has a close connection with the Data Preparation.
- Evaluation: High quality models (from a data analysis perspective) are selected. Before deployment it is important to evaluate the model and review the steps needed for its construction. At the end of this phase a decision on the use of the results is reached.
- Deployment: Knowledge gained in the previous phases is organized and presented in a way that it can be easily used. Depending on the requirements, the deployment can be as easy as generating a report or as complex as creating a repeatable process. It is important to understand what actions will be carried out to make use of the created models.

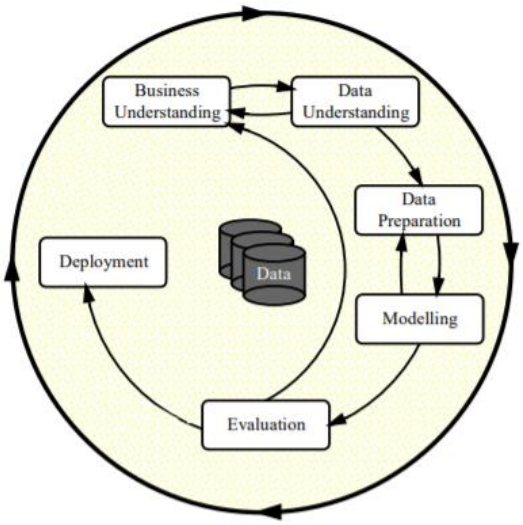


Figure 20: Phases of the Current CRISP-DM Process Model for Data Mining (Wirth, 2000)

The development of the system was separated in three phases:

1. The first phase consists of the implementation of multiple NN, testing each approach for evaluation indicators (accuracy, precision, recall, etc.). The Deep Learning Algorithms were trained using pre-existing datasets for the same effect of sentiment analysis based on social network data.
2. The second phase consists of evaluating the different models in newly collected data. These tests' results are compared via hypothesis tests or indicator comparison; The second phase includes a step for building a data

extraction system for social network data and stock data and feeding it to the selected algorithm in the first phase.

3. The third phase consists of the deployment of the model with the most favourable results. This deployment will use a real-world scenario to verify the functionality of the model.

4.3 System Architecture

The system architecture in Figure 21 displays the high-level interactions between the system components which include the database, the Deep Learning algorithms, and the Deep Learning Models

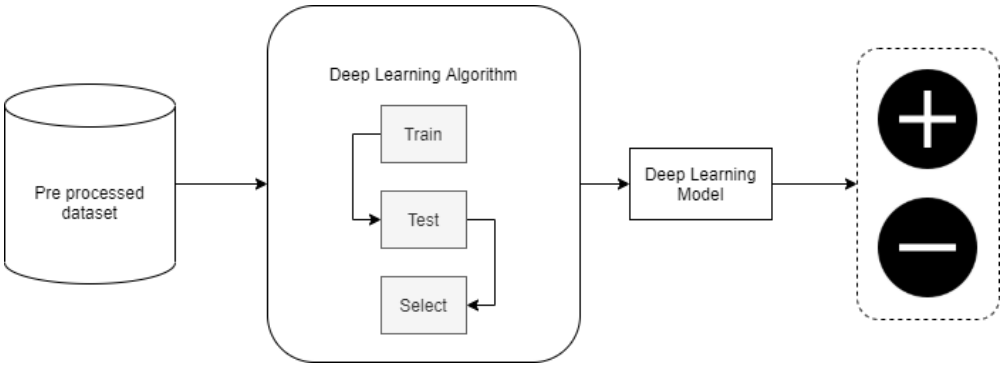


Figure 21: Pre-processed Dataset Architecture

For the Deep Learning Algorithm, Figure 22, shows how the different layers are stacked and communicate with each other. Keras is the top layer and the one with which the interaction will take place. Tensorflow is the backend which will be the interface between Keras and the dedicated architectures (CUDA, cuDNN, BLAS, Eigen, etc.) which communicate with the hardware and allow for parallelization.

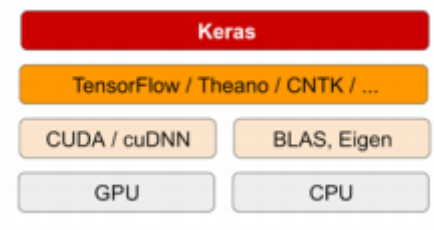


Figure 22:The deep-learning software stack (Ketkar, 2017)

4.3.1 Alternative Architectures

In this subsection are included the alternative architectures for the system, as well as alternative architectures for specific parts of the system, such as the Deep Neural Network, which will be tested in five different configurations.

Two different alternatives were planned in terms of the data pipelines. The first geared for the training phase, and the second for the test phase. Figure 23 depicts an alternative architecture from Figure 21 in which data is collected directly from the sources, instead of using a pre-existing dataset.

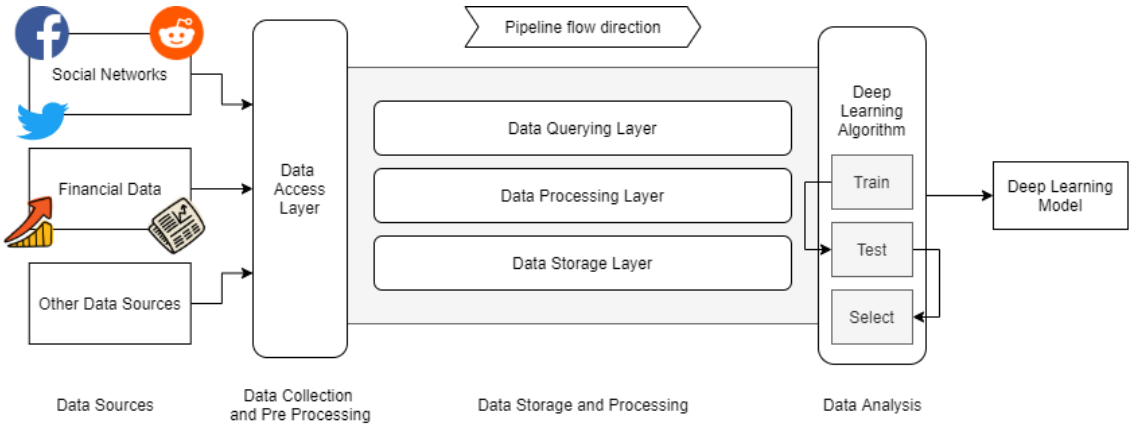


Figure 23: Data Pipeline Architecture

To understand how different Deep Learning methods deal with Sentiment analysis and which one is best geared towards this use case, seven different configurations will be reproduced.

Currently, in the state of the art, the most employed techniques for processing textual meaning in DNNs are CNN, LSTM and GRU. Each of these configurations deals with information in different ways. CNNs are good at extracting position invariant features, using pooling and fully connected layers. LSTM takes advantage of interconnected layers with several individual neurons. Each of these neurons is connected in a way that allows the network to retain, forget or expose important information. This structure allows the network to focus on the attributes with the most impact, maintaining a “memory” of them for the future. GRU is a subcategory of LSTM which does not contain a separate memory cell. An

LSTM cell controls the exposure of its memory to the other cells in the network, while the GRU always exposes all of it.

There are combinations of the different NNs mentioned above which work well together. To cover all the possibilities and understand which configuration is better suited for this use case, all three architectures are tested, CNN, LSTM, GRU, as well as possible combinations, CNN – GRU and CNN - LSTM. GRUs were also exploited in two distinct ways. Using a single Bidirectional GRU which, on the same layer, analyses data by the order at which it was input and one which reverses it. Or using two distinct layers, one which processes data in the input order and the other which reverses the input. From these configurations were created two algorithms, the CNN – GRU, which uses a single bidirectional layer and the CNN – BiGRU, which uses two distinct GRU layers.

A final NN was created with the purpose of relating sentiment data with numerical information. Given the use case, it is an opportunity to use stock data values. For this reason, the CNN – LSTM with Stock Indicators was also planned.

Another aspect which was reported to improve the performance of Deep Learning on text data was to use an attention mechanism. These mechanisms allow a NN to focus on certain aspects of the input, filtering out noise. Following the state of the art, the selected architectures to implement this feature were the GRU, CNN – LSTM, CNN – GRU and CNN – BiGRU.

The next subsections (4.3.1.1 through 4.3.1.5) depict the different NN configurations described in section 2.2 and their proposed architectures.

4.3.1.1 Convolutional Neural Network

In Rezaeinia *et al.* (2019) the DNN which was chosen for sentiment analysis was a CNN coupled with pre-trained word embeddings. To increase the accuracy of word embedding vectors, a combination of natural language processing, lexicon-based approaches, word position algorithm and Word2Vec/GloVe methods were all fed into the same CNN (Figure 24).

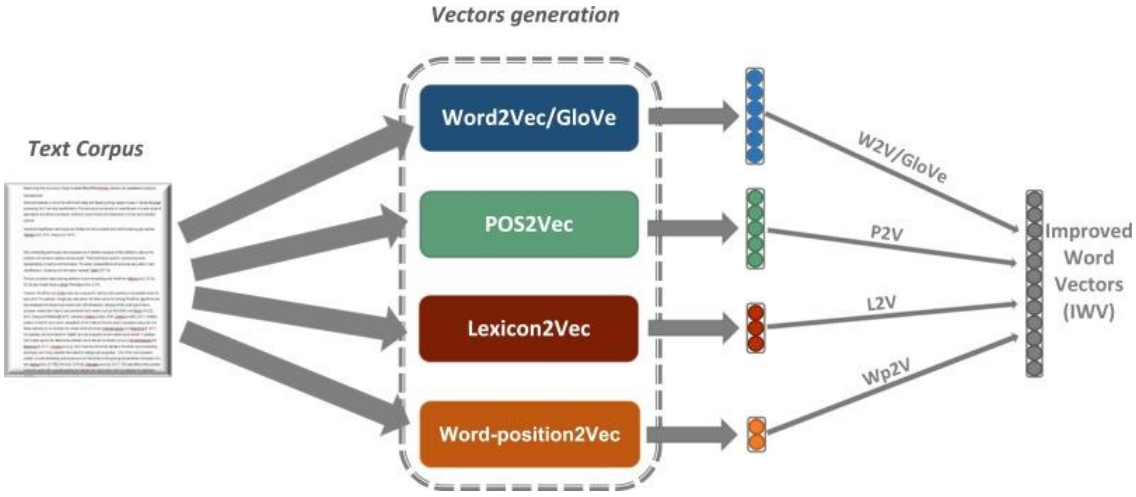


Figure 24: Improved word vector Architecture (Rezaeinia *et al.*, 2019)

The implemented CNN consists of 3 convolutional layers connecting to a Max pooling layer which in turn is connected to a fully connected layer. The result is a binary classifier (Figure 25).

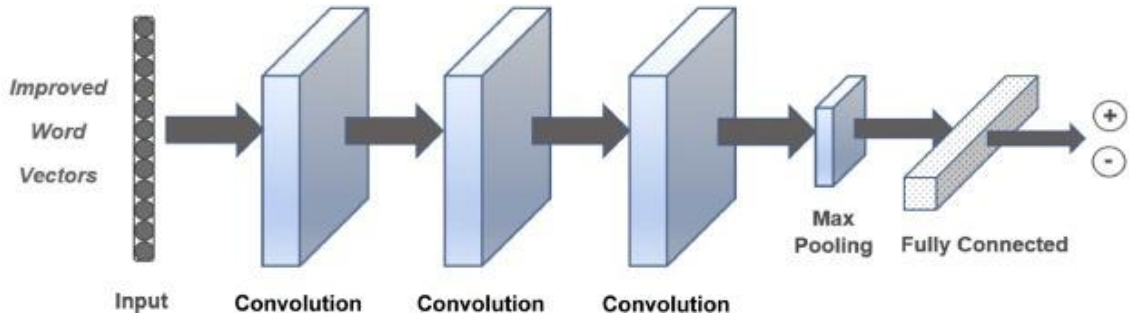


Figure 25: Improved word vector CNN Architecture (Rezaeinia *et al.*, 2019)

The implementation of a CNN in the scope of this dissertation will be based on this architecture and compared with the other alternatives.

4.3.1.2 Long Short-Term Memory

The LSTM architecture was based on the proposed LSTM-NN in Chatterjee *et al.* (2019). The architecture features a semantic word embedding and a sentiment word embedding which are then fed into the NN. The SoftMax layer's output consists of 4 classes – Others, Happy, Sad, and Angry (Figure 26).

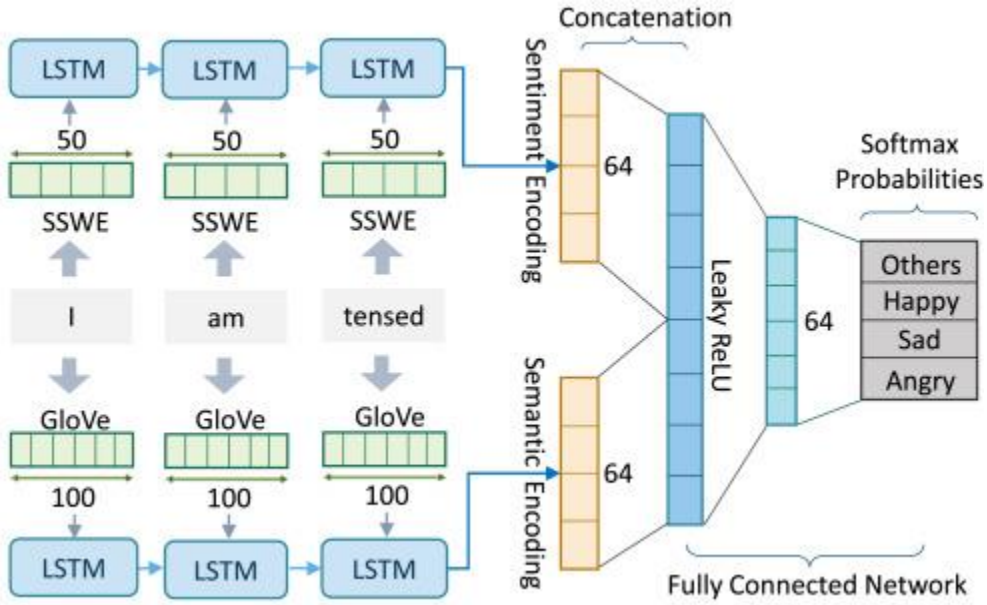


Figure 26:The architecture of Sentiment and Semantic Based Emotion Detector (Chatterjee *et al.*, 2019)

The implemented architecture for this dissertation's LSTM will pick up from the basis provided from the Figure 26 and change the SoftMax output two classes which correctly classify the stock market movements.

4.3.1.3 Gated Recurrent Unit

The basis for the GRU implemented in this dissertation is the proposed model in Yang *et al.* (2016) which includes a GRU-NN and an attention mechanism. The implemented NN is multi-

level. The first level consists of a GRU-based word encoder which feeds a word attention mechanism. The results of the word encoder and the word attention mechanism are then fed to a GRU-based sentence encoder which also feeds a sentence attention mechanism. The results of the sentence encoder and the attention mechanism are fed into a SoftMax layer (Figure 27).

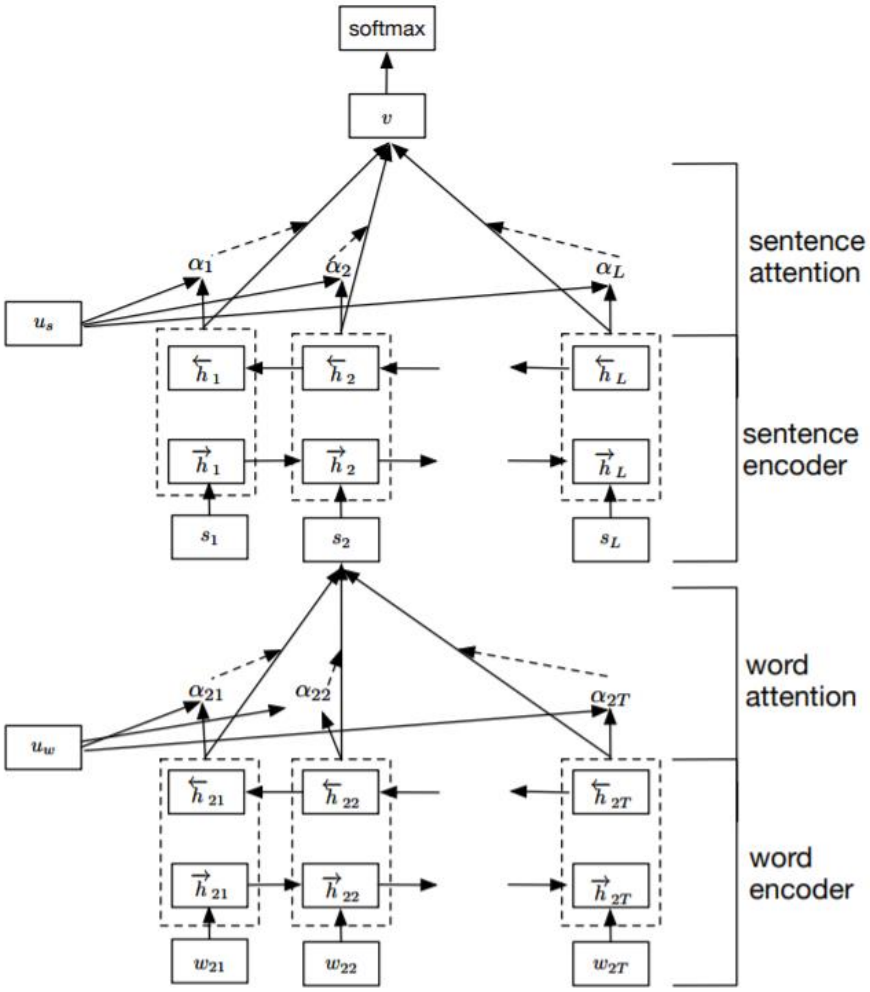


Figure 27: Hierarchical sentence network (Yang *et al.*, 2016)

4.3.1.4 Convolutional – Gated Recurrent Unit Neural Network

In Wen and Li (2018) there are two different hybrid GRU-CNN models, the Attention Recurrent Convolutional (ARC) model (Figure 28) and the Multiple Attention Recurrent Convolutional (M_ARC) model (Figure 29). Both models contain bidirectional GRU Networks which ultimately feed a CNN. The inputs of Bi_GRU are word vectors from embeddings.

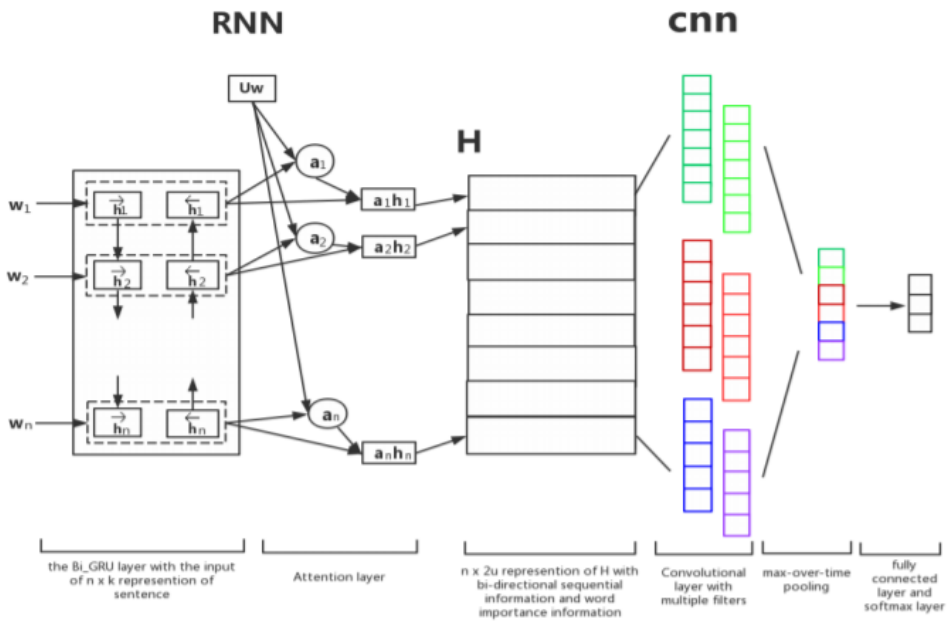


Figure 28: The Architecture of ARC Model (Wen and Li, 2018)

The main difference between the two models consists in the Attention Layer, the H matrix which feeds to the CNN and the configuration of the GRU Network. In M_ARC the attention layer is used to distribute forward and backward the information which allows the forward and backward hidden layers to get different importance distributions in two directions. The M_ARC model separates the GRU into two distinct feedforward and backpropagation networks, which feed Hf and Hb respectively. Convolution is carried on these two matrixes simultaneously.

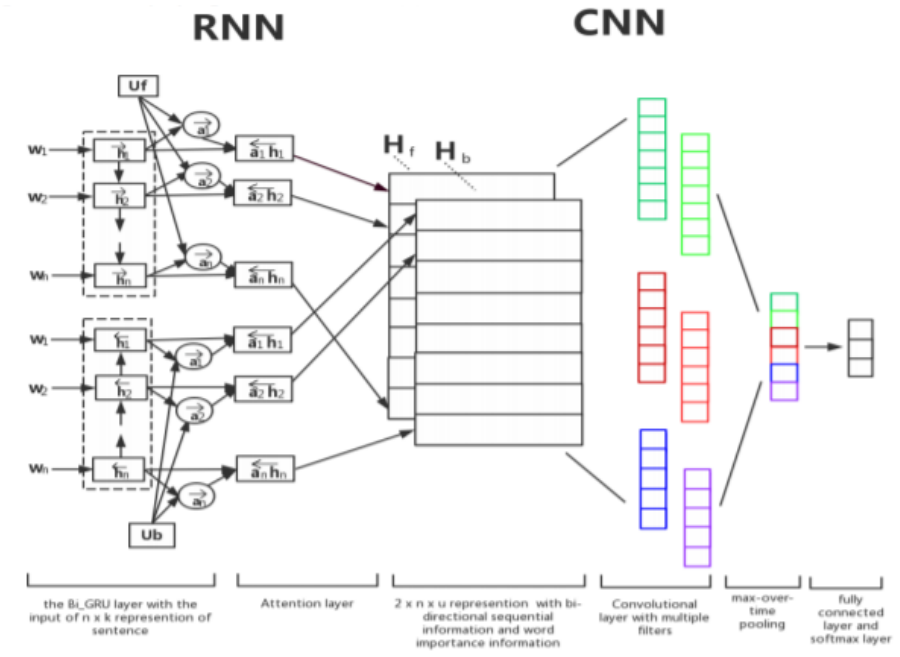


Figure 29: The Architecture of M_ARC Model (Wen and Li, 2018)

4.3.1.5 Convolutional - Long Short-Term Memory Hybrid Neural Network

The final Deep Learning architectures considered for this dissertation consist in a hybrid LSTM-CNN model. There are different implementations of such a DNN:

- Parallel LSTM and CNN networks.
- And the linear implementation of a CNN feeding an LSTM Network.

In Liu and Guo, 2019, the second scenario applies, consisting of a CNN which extracts information from text and feeds the Bidirectional LSTM layer with the results (Figure 30). Word embedding tables are used with pre trained word vectors. The CNN extracts the feature sequences which the BiLSTM processes, obtaining the preceding and succeeding contextual features. Two attention layers obtain the future context representation from the features extracted in the BiLSTM. Future and historic representations are combined to obtained and fed into a SoftMax layer. This architecture uses a loss function to update the Network parameters.

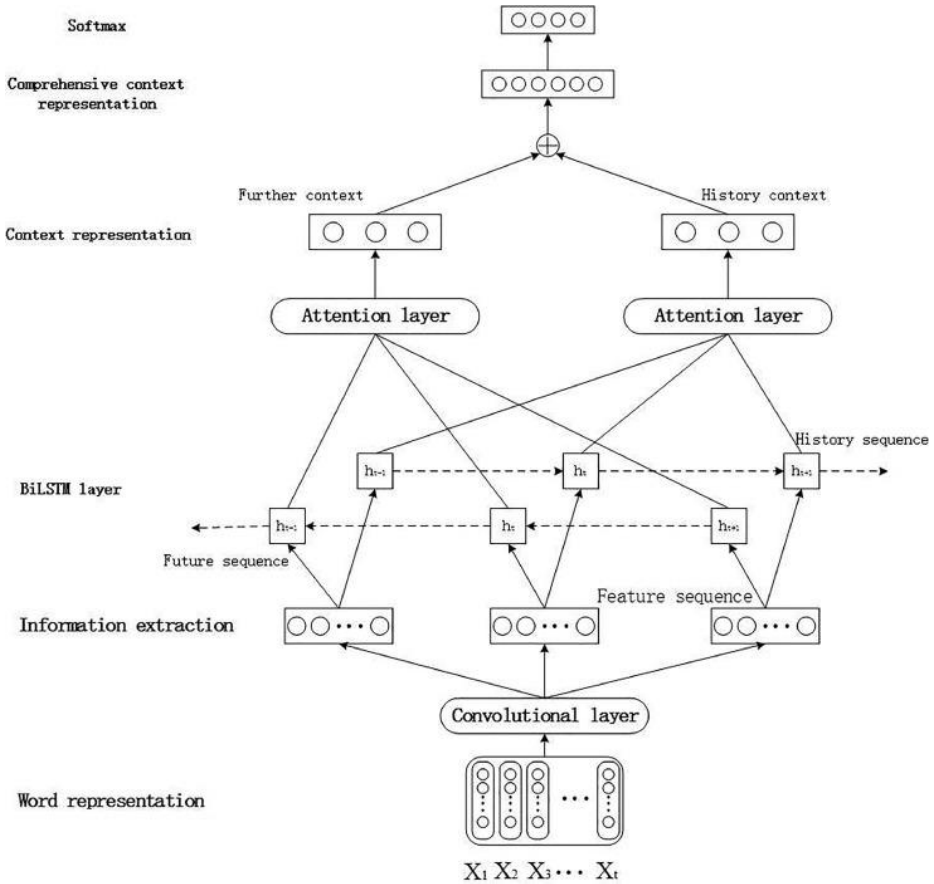


Figure 30: The architecture of the AC-BiLSTM (Liu and Guo, 2019)

An alternative implementation is to use parallel CNN and LSTM, which works well specifically with stock data, as the event embedding vectors can be fed into the CNN and the market variations into the LSTM. Event embedding vary from word embedding as the event embeddings separate sentences into actor, action, and object. Event embeddings can be generated by Neural Tensor Networks (Figure 31) which have word embeddings as input.

Event embeddings can be generated based on an average of its word embeddings, allowing the sharing of statistical strength between the words describing each component.

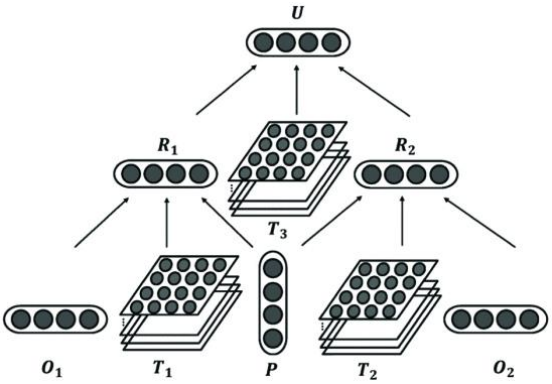


Figure 31: Neural Tensor Network (Oncharoen and Vateekul, 2018)

Using this kind of input, and stock market fluctuation values can result in a model such as the one in Figure 32 where events are embedded in time frames (months, weeks and days), allowing the Neural Tensor Network to produce the most important events for each of these time frames and feed them to the CNN for a broader overview of events over time.

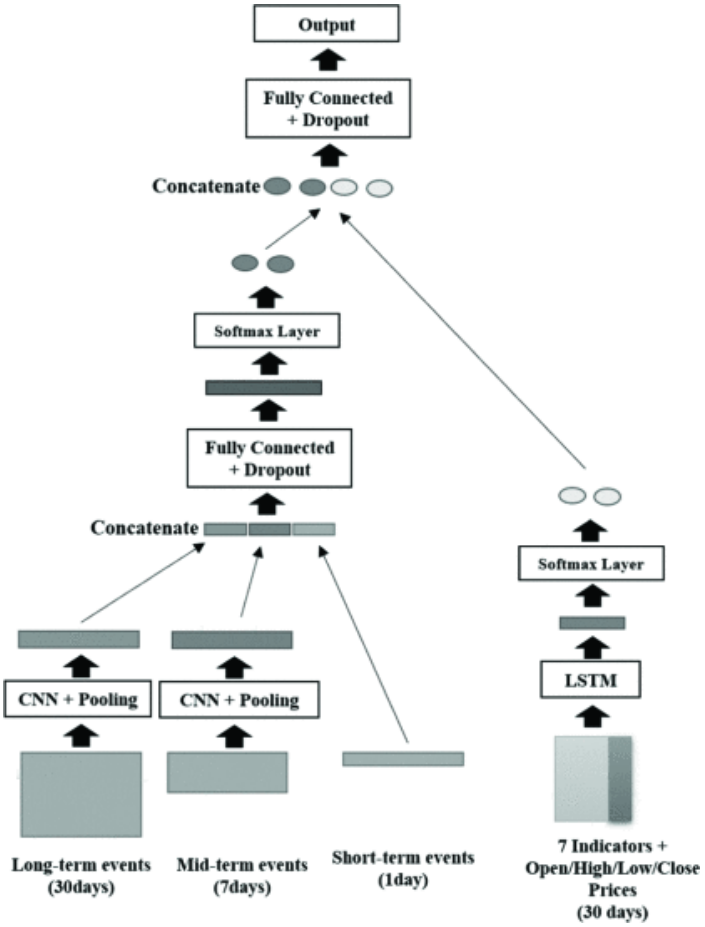


Figure 32: Parallel CNN - LSTM model (Oncharoen and Vateekul, 2018)

4.3.2 Data Sources

This section explains how the datasets used in this dissertation were obtained and provides their characterization. A single dataset containing Reddit News headlines with Dow Jones Industrial Average (DJIA) stock value information is used in the training phase. For the test phase AAPL, DJIA, BTC and TSLA data extracted from Twitter is used with their respective stock value information. Reddit datasets from three different subreddits is also used in the test phase. Stocks, Stock Market and Wallstreetbets, as well as a fourth dataset which combines the previous three, are the chosen data sources, each of them coupled with the DJIA stock value information.

In the training phase, it is preferable to use a pre-processed dataset. In this case one which was already used in other papers was used for convenience. will provide a baseline to compare the implementation to and eliminate noise which can occur in non-pre-processed datasets.

In Oncharoen and Vateekul (2018) there are three different datasets, one of which contains data from Reddit, a social network aggregator, which will be the one used for this comparison. The data consists of date marked headlines from 2008 to 2016, which have been already related to the DJIA which represents a price-weighted average of 30 significant stocks traded on the New York Stock Exchange and the NASDAQ, drops, and increases. Table 17 shows the results from the different models implemented (described in further detail in section 2.1.1).

Table 17: Accuracy on test data (Oncharoen and Vateekul, 2018)

Model	Accuracy
1. Event Embeddings	52.38%
2. News headlines and indicators	50.53%
3. News headlines (word embbeding) only	48.94%
4. Indicators only	51.85%
5. Event embeddings with indicators	50.26%

This dataset consists of three different tables as shown in Figure 33. A table containing DJIA information, with Date, Open, High, Low, Close, Volume and Adjusted Close for the DJIA index, with 1990 lines. Another containing Reddit news headlines, containing only Date and News columns, with 74377 lines. And the combined table has 1990 rows, each containing the date, the sentiment (1 for positive, 0 for negative) of the day and up to 25 different news headlines, extracted from reddit, for each date. This data will be transformed into a structure which will map each new headline to a single row, so there will be various rows for a single day, each with a single news headline.

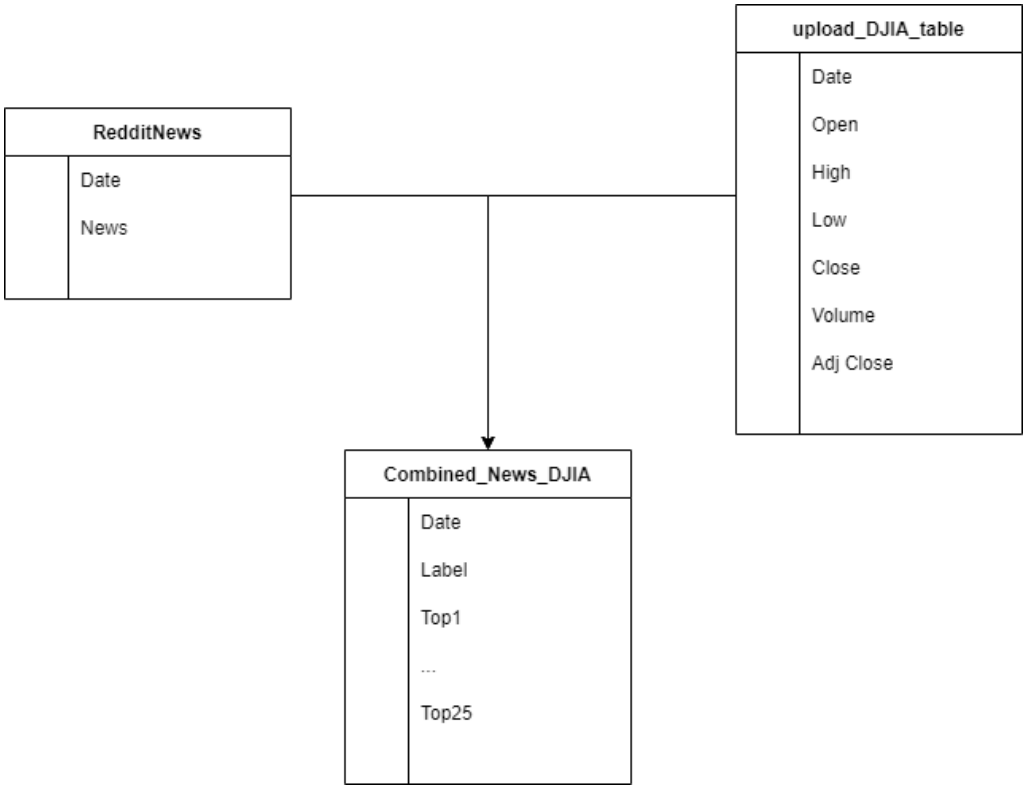


Figure 33: Training Dataset

For the alternative architecture in Figure 23, data extraction is necessary. The data sources are the Twitter API, the Reddit API and yahoo stock information. Figure 34 shows the table structure for the data which will used in the test phase. This data comes is first stored in three different tables and is then transformed into the two tables depicted in the bottom of the figure, to match the transformations which were also applied to the training data.

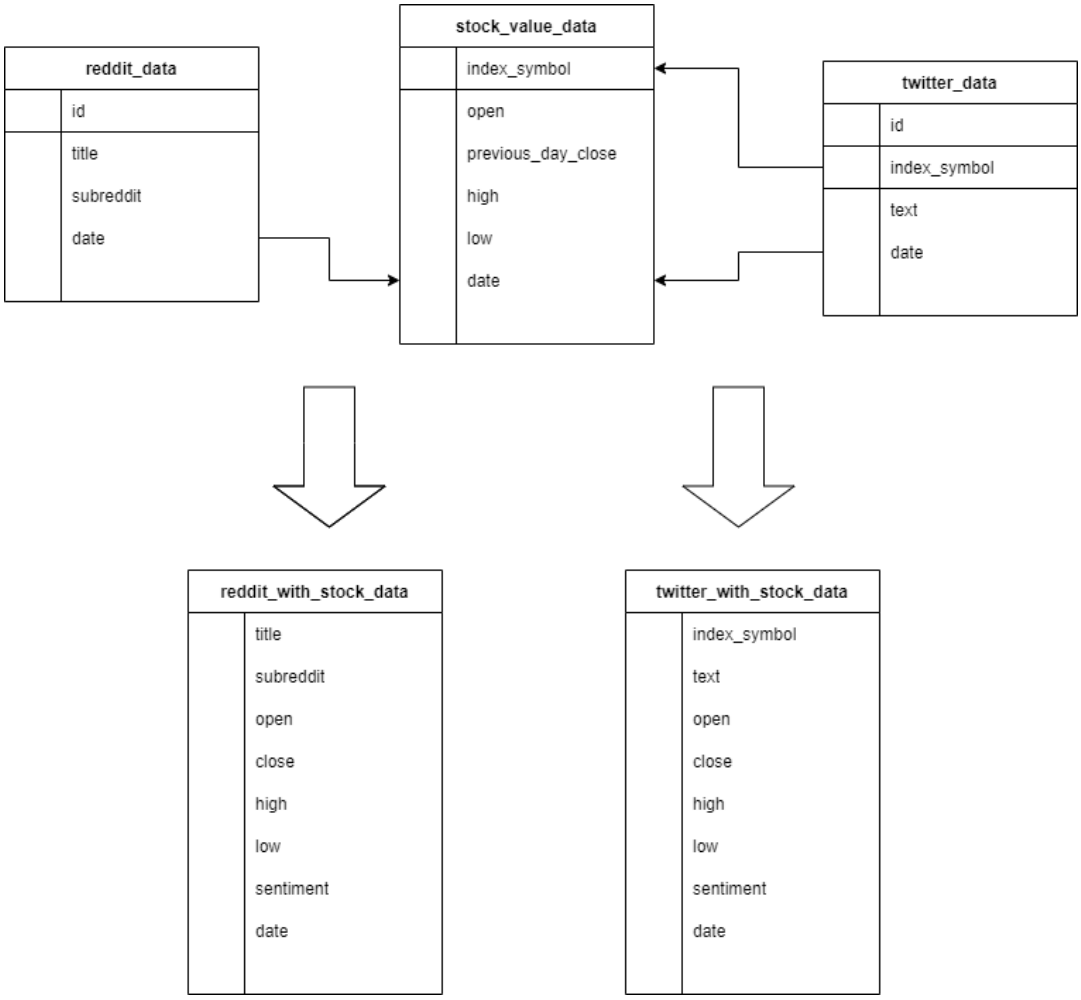


Figure 34: Test Dataset

The next section will focus on the implementation of the Deep Learning architectures mentioned above, as well as some experiments with each. The section will also focus on the development of the final solution using the selected DL architecture as well as the Big Data pipeline to complement it.

5 Implementation

This section covers the implementation of the distinct models, as well as the implementation of Deep Learning with a Big Data pipeline. The tried and tested architectures for DL are present in this section, as well as the selected one for the final implementation. The experiments done using the different DL algorithms, the results and the evaluation are detailed in the Evaluation chapter 6.

5.1 Development of Different Architectures

As mentioned in the previous chapter, the framework selected for the implementation of the Deep Learning algorithms is Keras. As this is a Python based framework, Jupyter Netbooks were used as a workbench for the development and testing of the different architectures, as it allows small snippets of code to be executed, which allows for debugging of the implemented code.

5.1.1 CNN based architecture

As mentioned in section 4.3.1.1, the implemented CNN architecture in Figure 35 was based on the Improved word vector Architecture (Rezaeinia *et al.*, 2019). The name Improved Word Vector comes from the fact that the input to a DNN is a tensor made up of various word embedding techniques, such as Natural Language Processing, Lexicon-Based approaches, word position, Word2Vec and GloVe methods. The Lexicon2Vec array, which is a Natural Language Processing technique which outputs the polarity of words, was implemented using six different lexicons as implemented in the original work (Rezaeinia *et al.*, 2019). The Part-Of-Speech Vector tagging, which consists of the context of each word in a phrase in function of its neighbours, was implemented using a Python framework and concatenated to the Word2Vec/GloVe array. The word position consists simply of a vector which contains the location of a word relative to the start and the end of a phrase. The Word2Vec/GloVe embedding was done using pre-existing bags of words mapped to arrays with size 300. This was done by mapping words to the correspondent array in word2vec, and then, if not present in word2vec, in GloVe. If neither of these dictionaries contained the word, a random array of size 300 was then generated.

The input for this model is a multi-dimensioned tensor which concatenates the embedding of each word, using the methods referred above. To make it possible for the Neural Network to process these inputs, each sentenced then had to be padded. This resulted in a tensor with a standard size for each dimension.

The Neural Network itself consisted of an input layer, followed by four convolutional layers (one more than the one employed in the original work) all with the Rectified Linear Unit (ReLU) activation. The next layer performs Max Pooling which pools the highest values from the convolutional layer output. A flatten layer reduces the dimensionality of the tensors in the network. Following are two Fully Connected layers, one with 100 units and a ReLU activation which is then fed to a simple Dense layer with only 2 units and a SoftMax activation. The output is a 2-dimensional array with the probability of a phrase being 0, or negative sentiment and 1, positive sentiment. The hyperparameters are not listed in the original paper, so there was some tunning involved to find the best ones, which consisted in testing different configurations and selecting the one with the best accuracy and loss results. Figure 35 contains a graph picturing the implementation, as well as the hyperparameters which were used in kernel and bias definition.

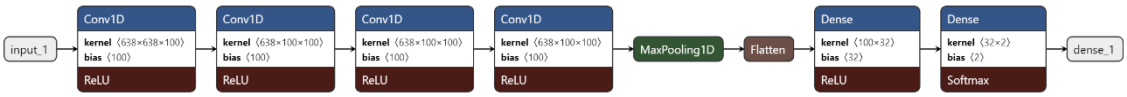


Figure 35: Implemented CNN

5.1.2 LSTM based architecture

The LSTM Neural Network implemented (Figure 36) was based on the on in section 4.3.1.2 and consisted of two distinct inputs, the first based on the GloVe embedding method which, as in the CNN architecture consists of arrays of 300 elements, as opposed to the one used in the original work which contains only 100 elements, the second based on semantic word embedding made up of arrays of 50 elements.

The constructed Neural Network is built with embedding layers which work as lookup tables, where each entry maps to an embedding array. To do this it is necessary to build a vocabulary, which is a map of numbers, each attributed to a different word. Only the 10000 most common words were included in the vocabulary. In the GloVe embedding layer, the embedding arrays for each word of the vocabulary are entered into a list and ordered according to the order of the vocabulary, this list is then assigned as the weights of the embedding layer, which cannot be trained as to not change the pre-assigned values. The SSWE embedding layer however is trained as it built as an empty layer.

Both branches of embedding are assigned their LSTM layer which connects to a fully connected Dense Layer. There is a need for a Global Average Pooling between the dense and LSTM layers due to the dimensionality of the data. The dense layer outputs are then concatenated and passed to two more fully connected dense layers, one of them a Leaky ReLU layer. The output layer is also a Dense layer which differs from the one implemented in Chatterjee *et al.* (2019), as it is a binary sigmoid activated layer, producing only the 0 and 1 for positive and negative sentiment respectively. There are also two dropout layers which were added to avoid overfitting of the model. The hyperparameters are not listed in the original paper, so there was some tunning involved to find the best ones, which consisted in testing

different configurations and selecting the one with the best accuracy and loss results. Figure 36 contains a graph picturing the implementation, as well as the hyperparameters which were used in kernel and bias definition.

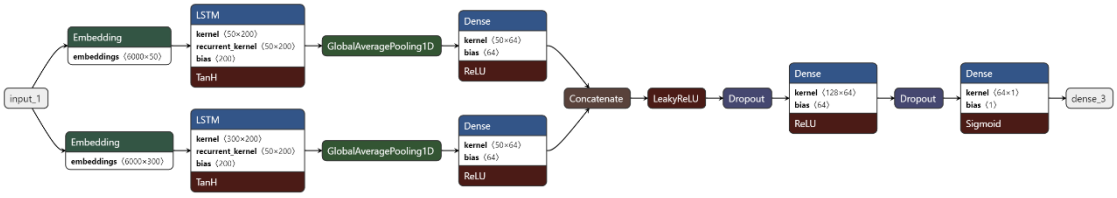


Figure 36: Implemented LSTM-NN

5.1.3 GRU based architecture

The GRU implementation shown in Figure 37 was based on the model in section 4.3.1.3, which also uses the Attention Layer. The input, much like the NN in the previous section, 5.1.2 LSTM based architecture, is obtained by embedding the words using an embedding layer with embedding arrays from GloVe. There is however a need for higher dimensionality, as there are two distinct encoding moments, one for words and the other for sentences, so, it was added a size 1 dimension level to the tensor to account for this. This architecture was developed for whole bodies of text and no single sentences as the ones in the dataset used for the exploration.

The Time Distributed layer was added to make it possible to analyse lower dimensions of the input. In this case, the word dimensions, and inside it is the embedding, the bidirectional word GRU and attention layers, including also a Dense fully connected layer with ReLU activation. The output of the Time Distributed layer is then passed to a similar network as the one it contains for sentence level encoding. There is also an addition of a dropout layer to avoid overfitting. The last Dense layer consists of a sigmoid, which results in a binary output, where 0 is negative sentiment and 1 is positive sentiment. The hyperparameters are not listed in the source paper and were obtained by tunning the model, which consisted in testing different configurations and selecting the one with the best accuracy and loss results. Figure 37 contains a graph picturing the implementation, as well as the hyperparameters which were used in kernel and bias definition.

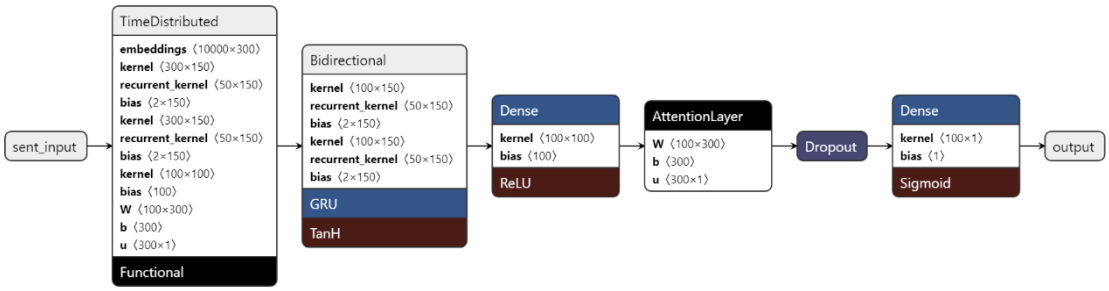


Figure 37: Implemented GRU-NN

5.1.4 Convolutional and GRU based architecture

Based on section 4.3.1.4, two Neural Networks were built

- one with a bidirectional GRU (Figure 38) (BiGRU)
- the other with separate forward and backward oriented GRUs (Figure 39), which changes the attention layer inputs and the number of attention layers needed.

There is a need for two in the separate GRU, where each will give attention to either the forward GRU or the backward GRU, and one only for the Bidirectional GRU which results in the attention mechanism being applied only to the bulk of the BiGRU outputs.

The embeddings, much as the previous architectures used the GloVe encoding method, which is referenced as one of the methods tried in the source paper (Wen and Li, 2018). The convolutional layers have been implemented with the ReLU activation function in both networks, and the GRUs have been implemented with TanH activation as was done in the paper Wen and Li (2018).

The output layer is a dense fully connected layer with a sigmoid output which outputs 1 if the sentiment is positive and 0 if it is negative. The hyperparameters are not featured in the source materials, so they have been selected by testing different configurations and selecting the best performers. Figure 38 and Figure 39 contain graphs picturing the implementation, as well as the hyperparameters which were used in kernel and bias definition of each model.

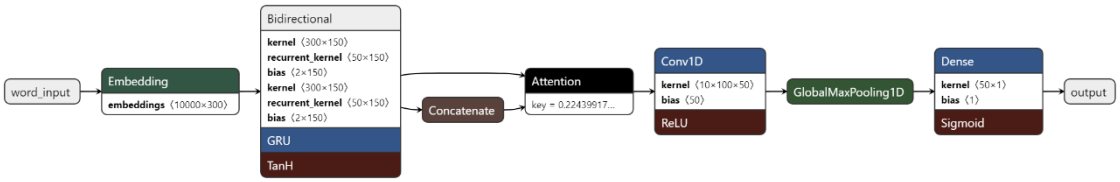


Figure 38: Implemented GRU-CNN with bidirectional GRU

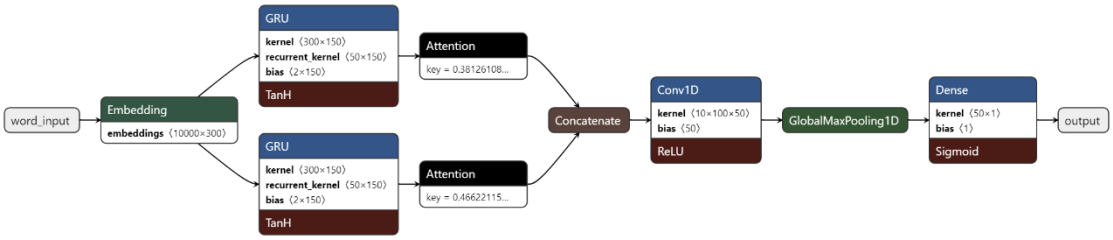


Figure 39: Implemented GRU-CNN with separated forward and backward encoding

5.1.5 Convolutional and LSTM based architecture

For the CNN and LSTM Neural Networks, two different concepts were tried:

- The first one, much like the previous implementations, consisted of using only the news data.
- The second approach was based on the work done in Oncharoen and Vateekul (2018) which uses numerical information in the form of stock values.

Figure 40 presents the first approach which uses an embedding method such as GloVe and passes the outputs to a convolutional layer. Next, data goes through a fully connected dense layer, and ultimately to a bidirectional LSTM (BiLSTM) which is connected to two different attention mechanisms for forward and backward LSTM processing. The output is a sigmoid dense fully connected layer, producing a binary result, 1 for positive sentiment and 0 for negative. This implementation is very much as described in Liu and Guo (2019). The source code does include the hyperparameter information, but further testing with different configurations resulted in the ones pictured in. Figure 40 contains a graph picturing the implementation, as well as the hyperparameters which were used in kernel and bias definition.

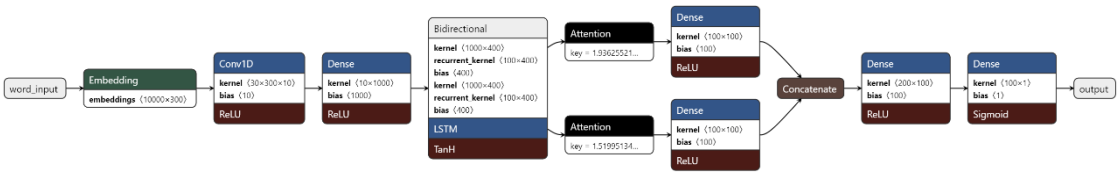


Figure 40: Implemented LSTM-CNN

Figure 41 contains the graph of the second approach, which uses the same dataset that was used in this paper for training all the implementations. Data consists of date marked headlines from 2008 to 2016 extracted from reddit. However, this approach changes the input format needed, as it contains three different inputs, one for each timeframe (month, week, day) of headline data and, there is an added input of the DJIA stock values (open, close, high, and low) as well as 7 different indicators based on these values which are used by stock traders in their investment strategies and is pictured in Table 18.

The implementation in Figure 41 shows the Neural Tensor Layer implementation, one for each time level (month, week, and day). After a global pooling layer, the NN feeds convolutional layers which are then concatenated. Another concatenation occurs with the result of the LSTM layer which processes the indicators in Table 18. Those indicators contain data for the last 30 days. The output of this neural network is binary, where 1 means there is a positive sentiment and 0 a negative one. The hyperparameters are defined according to the ones proposed in the work of Oncharoen and Vateekul (2018), as there was no significant variation in results with the different configurations which were tested. Due to the nature of this architecture being based on daily, weekly, and monthly data, a time-based approach was taken in training instead of randomly shuffling the data.

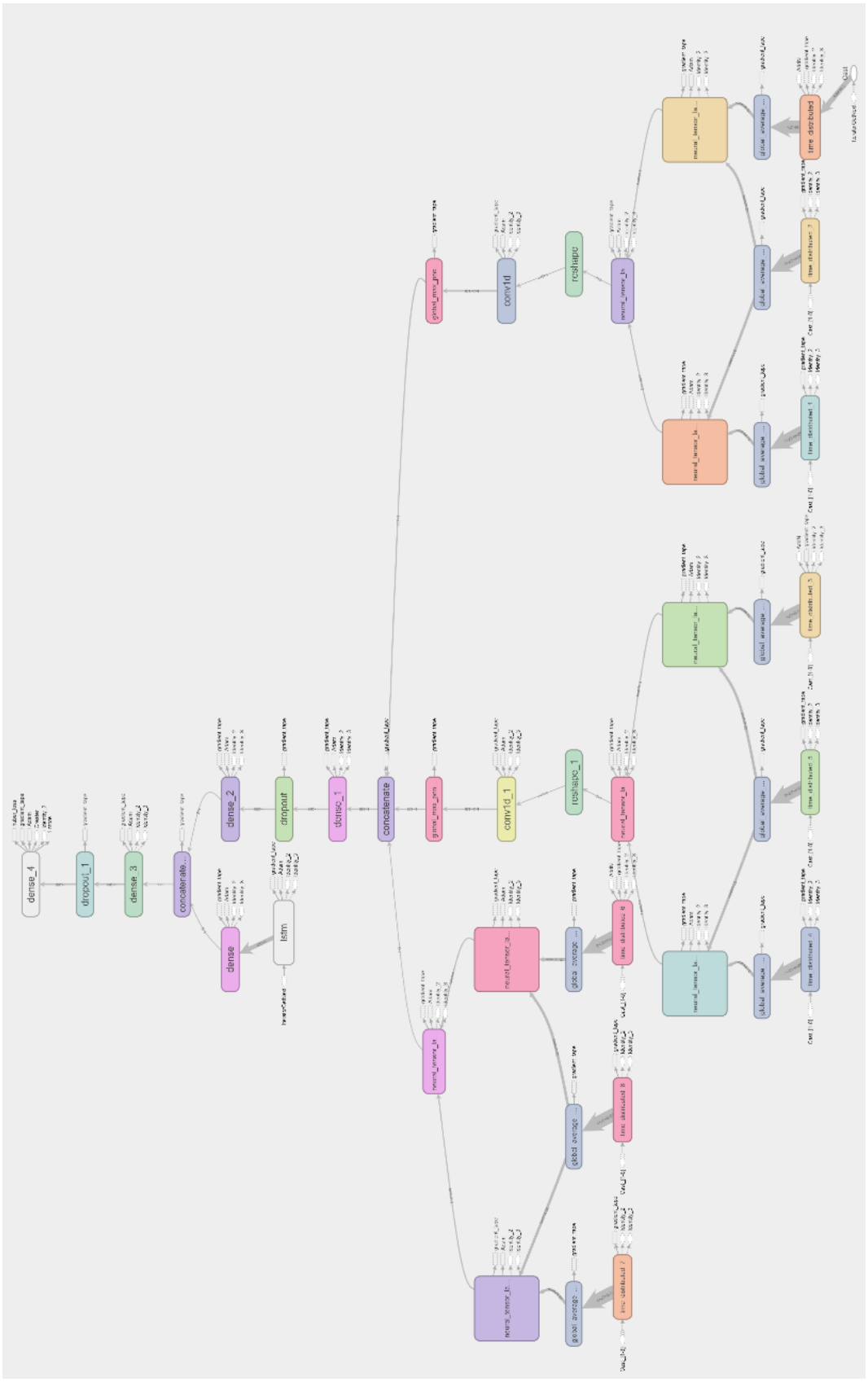


Figure 41: Implemented CNN - LSTM with Stock Indicators

Table 18: Stock Investment Indicators (Oncharoen and Vateekul, 2018)

Feature	Formula	Feature	Formula
Stochastic %K	$\frac{C_t - LL_n}{HH_n - LL_n}$	William's %R	$\frac{H_n - C_t}{H_n - L_n} \times 100$
Stochastic %D	$\frac{\sum_{i=0}^{n-1} \%K_{t-1}}{n}$	A/D Oscillator	$\frac{H_t - C_{t-1}}{H_t - L_t}$
Momentum	$C_t - C_{t-n}$	Disparity 5	$\frac{C_t}{MA_5} \times 100$
Rate of Change	$\frac{C_t}{C_{t-n}} \times 100$		

These architectures provided the foundation of the models which were to be used, but they needed to be trained on an already validated dataset. The next section focuses on the methodology for training these architectures.

5.2 Implementation Process

After implementing all the NN architectures, the next step is to train them, recurring to nested cross-validation (nCV), depicted in Figure 42. This approach consists of a cross-validation cycle (Handelman et al., 2019) inside another cross-validation cycle, where the inner cycle selects the model's best hyperparameters. These two loops work by dividing data into train and validation data, which the Tensorflow framework uses on its feedback and feedforward loops, providing a better fit during training. After the inner loop is finished, the resulting model is then used to classify the test data, which allows us to calculate the accuracy and loss of the model.

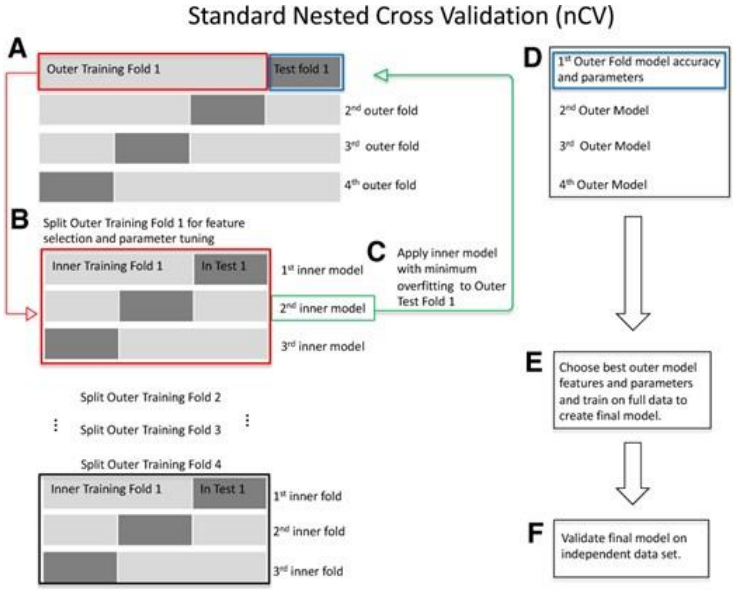


Figure 42: Nested Cross validation methodology (Parvandeh *et al.*, 2020)

The last model (LSTM-CNN with Stock Data), which uses stock data as well as sentiment data grouped by month, week and day as input was subject to a more specific training process. This model benefits from organized data, due to the way its input is organized, in this case a time series. This meant that a specific nCV approach was used, where, instead of looping through all data randomly, it starts by the first date of the dataset and increasingly it is allowed more data as it progresses. This method allows the model to learn that there is a structure in the data itself Figure 43. Such as nCV, this time series approach divides data into test, train, and validation, where the inner loop has access to the test and train data and the outer loop outputs the accuracy of the model. The validation data becomes part of the train data in subsequent runs and the test data part of validation, while test data is always never seen before data, as it is the most up to date.

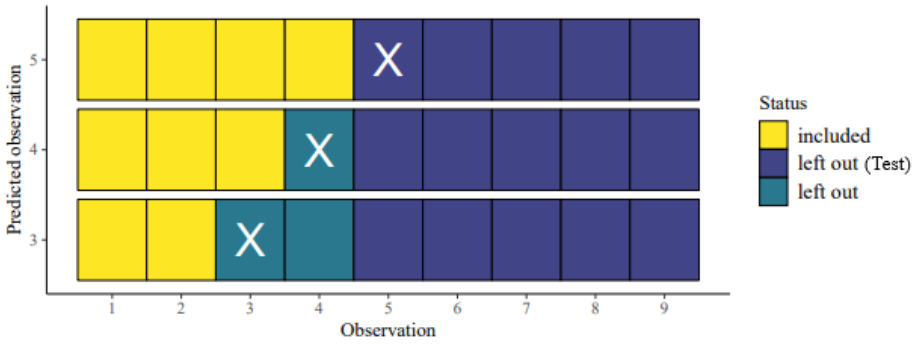


Figure 43: Nested cross-validation for a time series (Bürkner, Gabry and Vehtari, 2020)

After the training, the resulting models are then going to be used on data collected from other sources. This will allow us to compare not only the training results of each model, but also the real world usability of each.

The next section explains how data was gathered and stored prior to being used to test the afforementioned models.

5.3 Data Collection and Storage

As mentioned in previous sections, the Deep Learning models have been trained using a curated dataset, containing Reddit news headlines with the sentiment value, which is inferred from the difference between close and open values of the stock markets for the DJIA index, from Oncharoen and Vateekul, 2018. This dataset was already used on other research papers, so there was no need to collect and prepare any data for the training phase, as it arrived in a text format, ready for use. The test phase however will rely on data collected from Social Networks and Stock Data trackers.

For the test phase, in which there is an in-depth performance comparison of the different models produced in the previous section (5.2), external sources will be used, mainly

Twitter, Reddit and the Yahoo! Stock API. The objective is to understand how each model reacts to different scenarios. The information consists of posts from different users with questions, advice, etc. Twitter data was obtained using filter terms, which resulted in data for the Apple (AAPL), Bitcoin (BTC-USD), Dow Jones Industrial Average (DJIA or ^DJI), and Tesla (TSLA) indexes

Using Big Data technologies, such as Spark, Hadoop/HDFS and Hive, it was possible to implement a data warehouse. Data was extracted from Twitter and Reddit in an hourly basis and the Yahoo! Stocks API produced the daily stock values for four different indexes (DJIA, BTC, AAPL and TSLA). In Figure 44 we can see the resulting architecture for the Big Data storage. Three different applications query three different APIs (Twitter, Reddit, and Yahoo! Finance) and write the resulting responses to the HDFS cluster using Spark as the engine for the write operation. Hive stores the metadata of the tables in HDFS in a MySQL database, which allows it to keep track of tables, columns, etc. Spark acts as the engine for Hive as well. Hive queries are executed within Spark and their result then sent back to Hive. These queries help to update the metadata tables in MySQL.

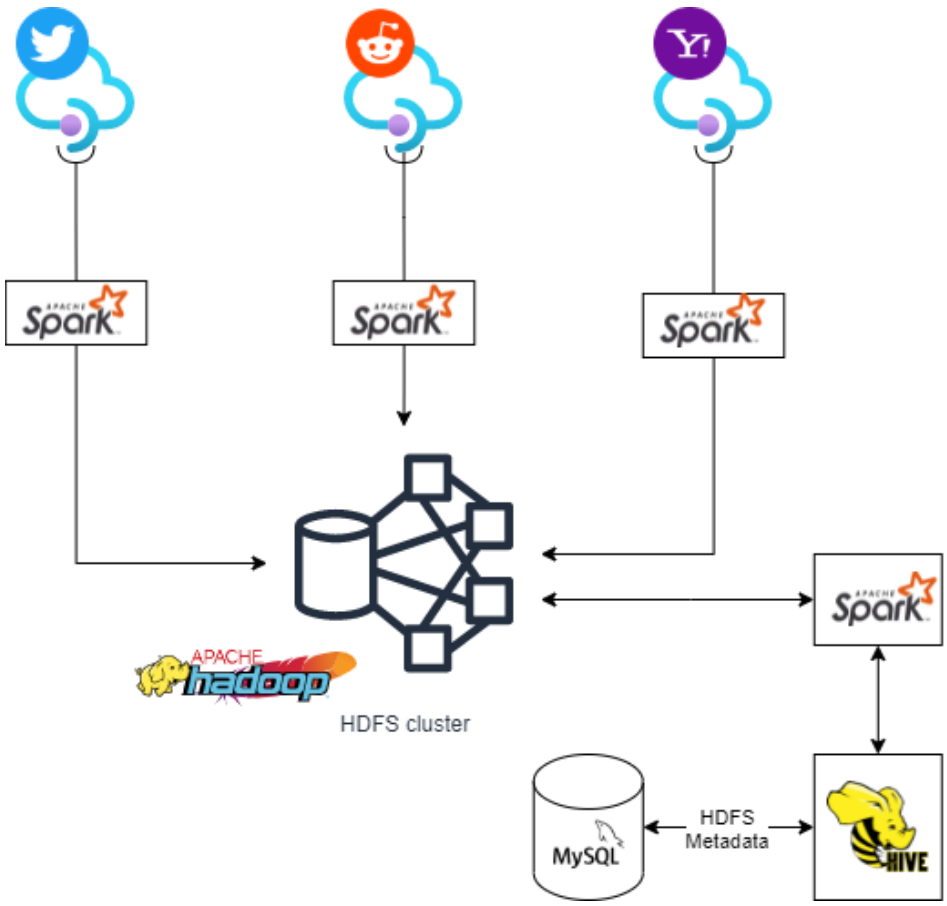


Figure 44: Big Data Storage Architecture

The API data was stored in three different tables, as pictured in Figure 34. Reddit data is stored with the id, title, subreddit and date on which the information was written. Three different subreddits were selected for data extraction, all of them related to stock data (stocks,

stock markets, and wallstreetbets). For Twitter, three different terms were used to filter tweets. Each table entry contains the date at which the tweet was written as well as the Stock Index name it relates to. Stock Value data from Yahoo! Finance contains the end of day values for the same indexes which were filtered in Twitter data.

Reddit data concerns general inputs over stocks, much like de training dataset did. As DJIA is an aggregation of various stock market indexes, it is used to relate to the title data. The sentiment is then calculated using the following rule: sentiment is positive if the stock close value is higher than the stock open value, otherwise it is negative. The same rule is also applied to the Twitter data when joined to each of the stock index values. Since we have the values for the stock indexes filtered in Twitter, we do not generalize by using DJIA values and associate each tweet to the index it represents.

Data from the tables in is divided into 6 different datasets for testing the Deep Learning models. Table 19 explains each of the datasets in terms of number of lines, number of days and origin. All Stock indicator data is in the table *stock_value_data* and is joined with the data from each dataset. The resulting datasets from Twitter and Reddit are then filtered, in Twitter's case, by the stock index (AAPL, BTC, DJIA, TSLA), and in Reddit's case by the subreddit (Stocks, Stock Market, Wallstreetbets), with a fourth dataset keeping all subreddit data together.

Due to the way the stock market works, Table 19 contains only 12 days for the AAPL, DJIA and TSLA datasets, while BTC has 18 days. The explanation is that the stock market only allows trades during workdays, while BTC, a cryptocurrency can be traded any day. Reddit data was collected earlier, but not all subreddits had posts for every day.

Table 19: Collected dataset description

Dataset	**AAPL**	**BTC**	**DJIA**	**TSLA**	**Stocks w/DJIA**	**Stock market w/DJIA**	**Wallstreetbets w/DJIA**	**All Reddit data w/DJIA**
Number of senteces	21936	14600	12640	22800	21132	21203	20270	62605
Number of days	12	18	12	12	25	28	15	28
Origin	Twitter	Twitter	Twitter	Twitter	Reddit	Reddit	Reddit	Reddit

After covering the implementation of the different algorithms, the process that was used for training each of the models and the process of data collection for testing each produced model, the next section covers experimentation and analysis. Using the outputs produced in this phase, metrics were collected and compared to understand how well each model behaves in different contexts.

6 Evaluation and Experimentation

This section focuses on evaluating the developed system by setting indicators and metrics, as well as identifying the information sources used to generate results. The hypothesis is first defined. Lastly, the methodology for the evaluation is explained.

6.1 Objective

The objective of this thesis is to understand if Deep Learning is a good analytic method to extract value from Big Data, circumventing the troubles created by the volume, velocity, variety, and veracity associated to it. This is achieved by training and testing different Deep Learning models on data which has the properties of Big Data (Volume, Velocity, Veracity, Variety, Value). To make sure the results obtained are representative and have statistical significance, different hypotheses were formulated for the different metrics which were collected. After confirming the results have statistical significance, it is possible to take the appropriate conclusions and select the models which produced the best results.

The next section elaborates which indicators can be used to validate this hypothesis as well as their provenance.

6.2 Indicators and Information Sources

ML has a set of core performance indicators which can be used for a model's self-evaluation. These can be extracted from a confusion matrix (Figure 45). A confusion matrix is a way to record and extract meaning from predictions and true values. False positives (FP) are values predicted to be positive but are actually negative. False negatives (FN) are values predicted to be negative but are actually positive. True positives (TP) are predicted and actually positive. True negatives (TN) are predicted and actually negative.

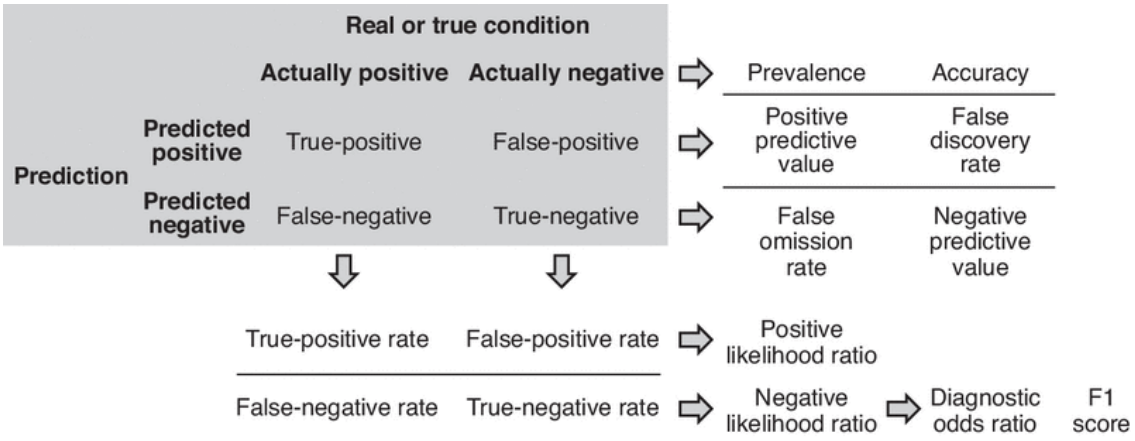


Figure 45: Confusion matrix (Handelman *et al.*, 2019)

These metrics allow the extraction of indicators, such as accuracy (1), or the ratio of correct predictions to total predictions made,

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

(1)

precision (positive predictive value) (2) or the proportion of positive identifications which was actually correct,

$$P = \frac{TP}{TP + FP}$$

(2)

recall or sensitivity (true-positive rate) (3) or the proportion of actual positives which was correctly identified,

$$S = \frac{TP}{TP + FN}$$

(3)

fallout (false-positive rate) (4) or the ratio between the number of negative events wrongly categorized as positive (false positives) and the total number of actual negative events,

$$F = \frac{FP}{FP + TN}$$

(4)

specificity (true-negative rate) (5) or proportion of negatives that are correctly identified as such,

$$SP = \frac{TN}{TN + FP}$$

(5)

F1 Score (harmonic mean of precision and sensitivity) (6),

$$F1 = \frac{2TP}{2TP + FP + FN} \quad (6)$$

and many others (Handelman *et al.*, 2019) .

Matthews Correlation Coefficient (MCC) (7) is a performance metric which focuses on all four quadrants of a confusion matrix. May be advantageous as it only rewards models when they have good performance in all quadrants. MCC produces values between -1 and 1, where 1 shows complete agreement of correlation and -1 complete disagreement. If the result is 0, the prediction is said to be uncorrelated with the ground truth.

$$MCC(\theta) = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(FP + FN)(TN + FP)(TN + FN)}} \quad (7)$$

Other elements such as time to train and hardware resource usage are also considered important in this dissertation. Deep Learning was used for Big Data as traditional ML algorithms would be used in traditional datasets. A local computer was used to store, process, consume, access, and analyse data, and its details are listed in Table 20. The hardware usage and time of training are collected during the training of the models by logging and profiling the resource usage.

Table 20: Training Computer Specification

Specification	**CPU**	**GPU**	**Memory**	**Storage Capacity**
Description	AMD Ryzen 5 3600 – 3.6 GHz	NVIDIA RTX3070 8GB vRAM	16GB	4TB

The next section illustrates the methods which were used to obtain the indicators above.

6.3 Evaluation Methodology

The evaluation is made with a basis on the indicators defined in the previous section. This evaluation is divided in three different stages, as shown in Figure 46. The training phase will output the accuracy, loss, and training time of each algorithm. In this phase, only the Reddit Stock Headline dataset is used, as it has already been pre-processed.

The second phase consists of the evaluation of the models with different test data. These evaluations output the accuracy, precision, sensitivity, fallout, specificity, f1 score and MCC. The datasets used in this phase are the AAPL, BTC, DJIA and TSLA created from Twitter

data, as well as the Stocks, Stock Market, Wallstreetbets from Reddit, plus an extra one with a mix of all Reddit data.

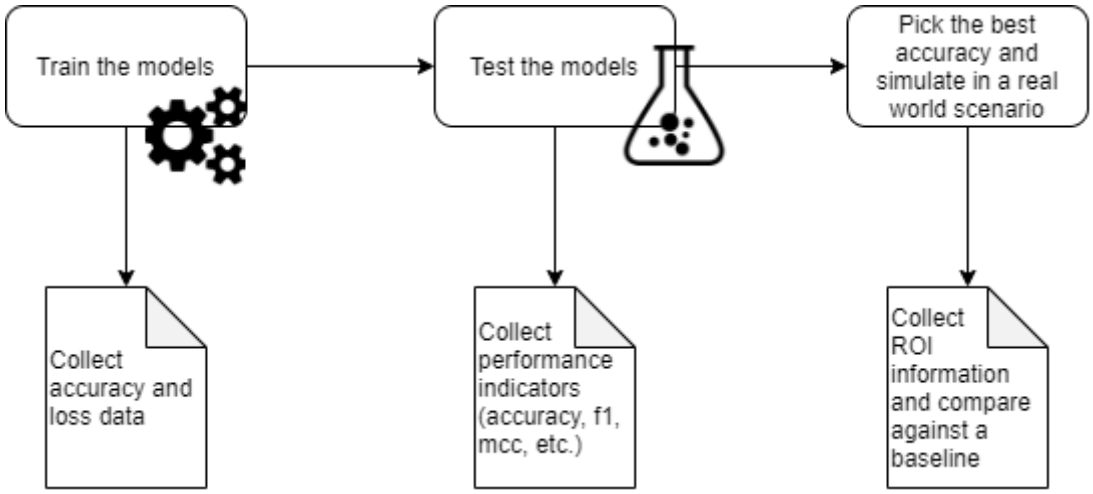


Figure 46: Experimentation Methodology

Since the example used in this dissertation concerns stock data, for each trained model, an investment simulation was added. After finding out the 2 best performing models in the test data, these are used to simulate an investment strategy. The investment strategy consists of starting with 1000 dollars which are invested in the days when the model predicts a positive sentiment. When there is money invested, if the models predict a negative sentiment for a certain day, all the money is extracted to “savings”. When a positive sentiment is predicted, all the money in “savings” is invested, despite there being a loss or an earning compared to the initial investment.

All the models except the CNN - LSTM with Stock indicators evaluate the sentiment of phrases in relation to the stock market individually. CNN – LSTM with Stock indicators evaluates the aggregate sentences for a whole day, as well as the past sentences to that day, so it is the only model ready for the real-world trading simulation. For the other models, in case they end up being selected, the simulation is done by averaging the predicted sentiment of all sentences. Since sentiment is a seen as a binary value, if the average is higher than 0.5, then it is considered as positive sentiment, otherwise it is considered negative.

In the next section, the detailed methodology is followed, and the results are presented.

6.4 Results and Analysis

The focus in this section is to present the data collected in the experimentation phase and to expose it in a convenient way so that conclusions can be drawn upon it.

The three different experimentation phases are presented, starting by the data collected on the Deep Learning models training phase, in which different configurations are trained with the same dataset. The test phase, in which eight different datasets were

generated after collecting, transforming, and readying the data for processing by the models generated in the test phase. The real-world simulation consists of simulating a trading strategy aided by the results of the different algorithms.

6.4.1 Training

The training phase yielded the accuracy and loss of the different models in Table 21. The values collected were done so by averaging each of the accuracies on test data from the nested cross-validation method of training.

Table 21: Training indicators

Indicator	CNN	LSTM	GRU	CNN - GRU	CNN - BiGRU	CNN - LSTM	CNN - LSTM with Stock indicators
Accuracy	0.520	0.644	0.534	0.632	0.591	0.534	0.726
Loss	0.125	0.073	0.025	0.105	0.093	0.233	0.125
Training Time	47 hours	14 hours	22 hours	16 hours	17 hours	22 hours	20 days

The CNN algorithm trained for 47 hours, as depicted in Figure 47, where the green line represents the accuracy results on validation data, while the pink line represents the accuracy on training data. It seems that the variation in train accuracy is not very high, although it fell over time, ending below 50%. Test accuracy variated between 54% and 47%. The average result on the test data was 52%.

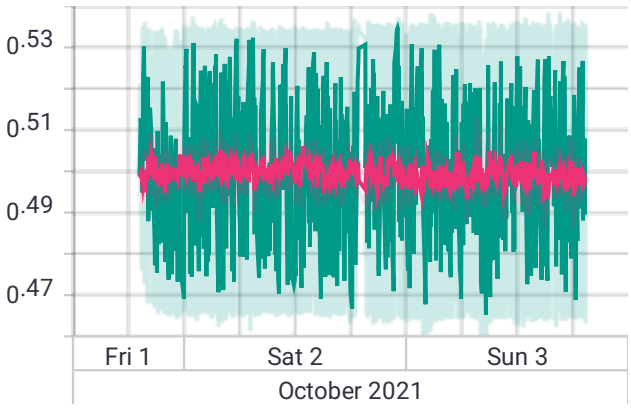


Figure 47: CNN train accuracy

The loss value changed very little during training. Figure 48 shows no green line as there was no observable difference between loss on train data and loss on validation data. This may be a negative sign, and maybe evidence that something went wrong during training, possibly underfitting. To solve this issue, more different hyperparameters could have been tested.

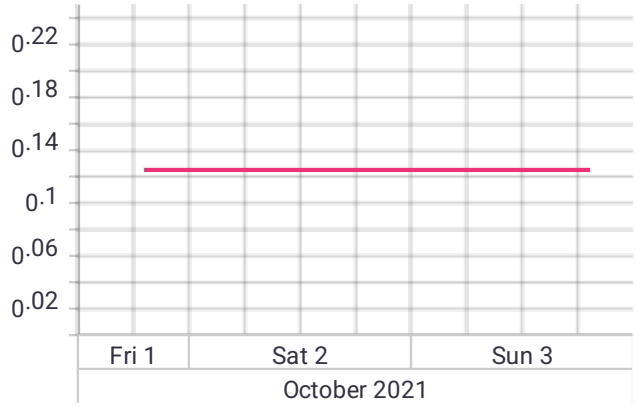


Figure 48: CNN train loss

This algorithm also kept a regular usage of the GPU RAM resources, keeping its usage at about 1.5GB. Figure 49 shows the evolution of the use of memory by CNN.

Figure 49: CNN memory profile

The LSTM algorithm started with a high accuracy, both on training and in validation, reaching 90% on training and 85% on validation. It then started decreasing, ending on around 85% in training data and in 80% on validation data. The results on test data ended up with a 64% accuracy score. The training curve is depicted in Figure 50 where the blue line is accuracy in validation data and the red line is the accuracy in train data. With a training time of 14 hours this ended up being the algorithm with the fastest training time.

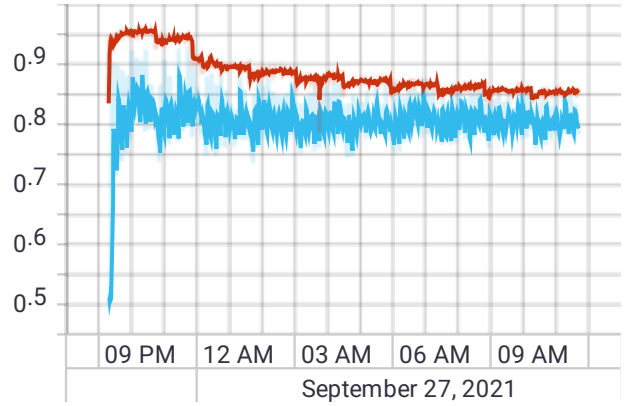


Figure 50: LSTM train accuracy

Figure 51 shows the loss curve during the training of this model. It seems to be a mirror image of the accuracy, starting very low and rising through time. In the end, the loss value in test data was around 7%, the second best. The colour of the lines matches the colours in the accuracy curve. Both the accuracy and loss curves seem to indicate that there is stability in the validation and the training curves.

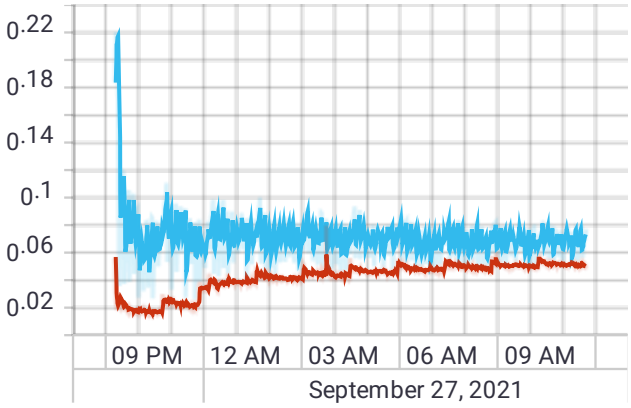


Figure 51: LSTM train loss

The memory profile from Figure 52 shows that the heap usage was somewhat volatile, switching between the high 4GB and low 5GB of usage.

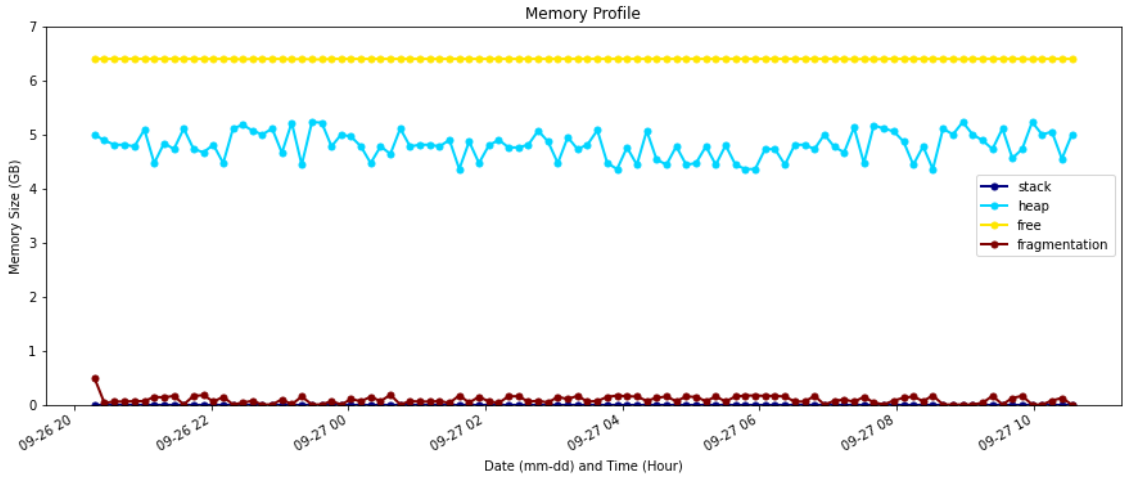


Figure 52: LSTM memory profile

Figure 53 shows a similar behaviour between training in LSTM and GRU in the beginning. However, when LSTM started dropping it accuracy, GRU stabilized. Figure 53 shows the evolution, with a red line for accuracy on training data and blue with accuracy for validation data. The accuracy on test data ended up being 53%.

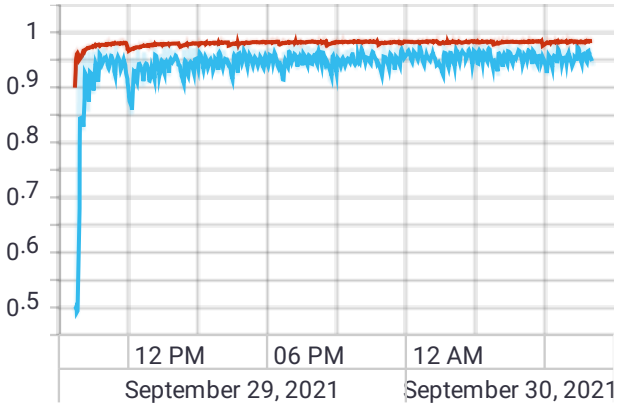


Figure 53: GRU train accuracy

In Figure 54 there is a loss curve depiction with the same colour code as the accuracy curve o. It stabilized near 0 for training data, the line in red. The loss achieved by this algorithm was 2.5%, the lowest of the samples. Both the training and validation curves seem to stabilize which indicates a good fit on data.

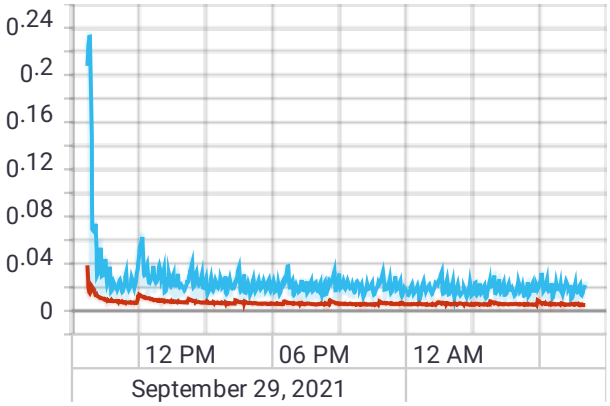


Figure 54: GRU train loss

Figure 55 shows a stable use of about little less than 5GB of GPU memory. This memory profile is kept stable during the 22 hours it takes the algorithm to train.

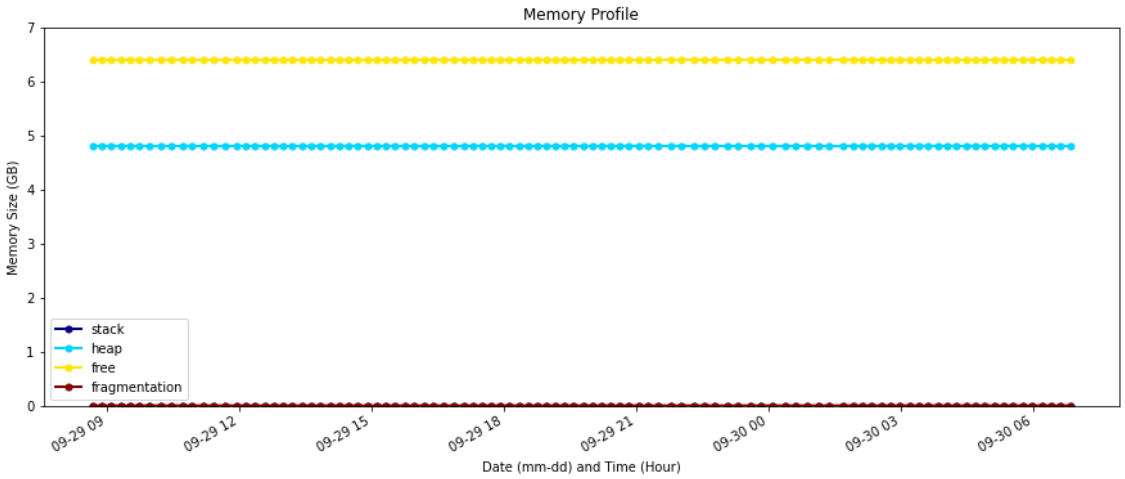


Figure 55: GRU memory profile

Figure 56 shows an interesting pattern where the accuracy (in pink) seems to be stable until it drops. Validation (in green) seems to keep up with the changes in accuracy. The final accuracy on test data finished in 53%, which makes sense given the interval in which accuracy varied in train and validation data.

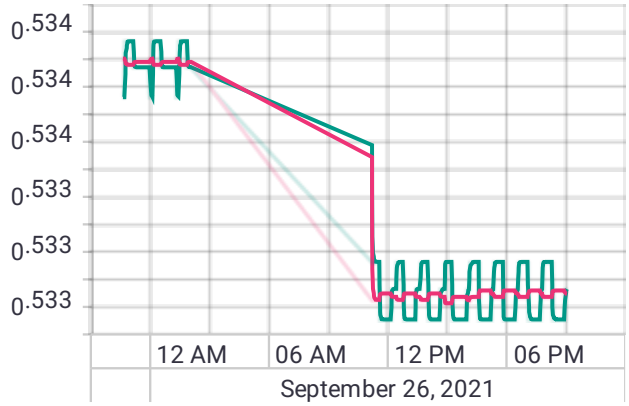


Figure 56: CNN – LSTM train accuracy

Figure 57 shows that loss kept the same values for both validation and train data, causing the green line to not be visible. It also seems that loss suddenly increased in the same way that accuracy fell. Loss was 23% in the end.

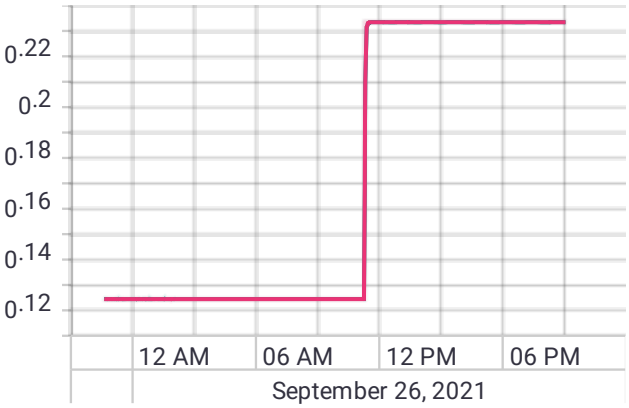


Figure 57: CNN - LSTM train loss

In terms of resource usage, represented in Figure 58, it also kept stable near 5GB of GPU heap memory usage. This algorithm took 22 hours to train.

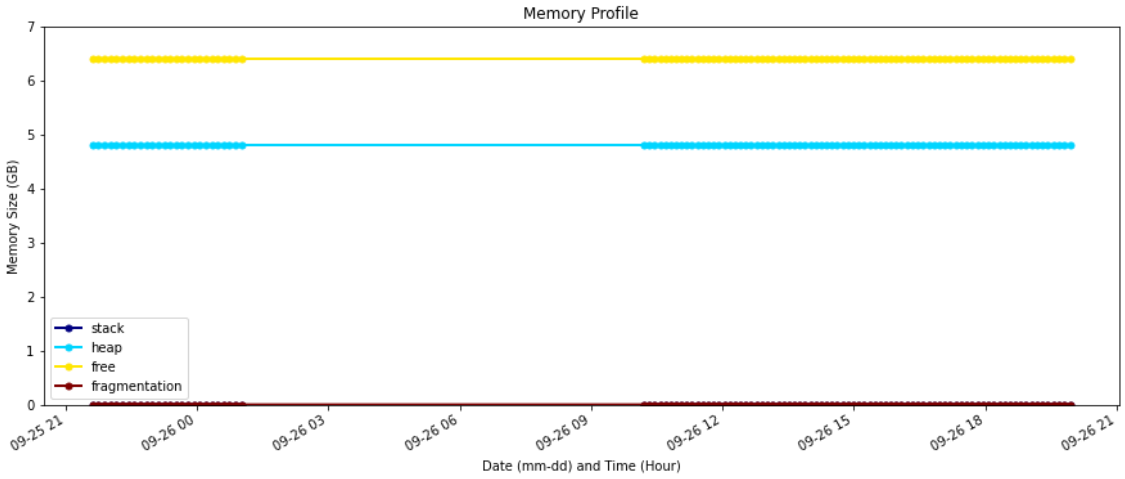


Figure 58: CNN - LSTM memory profile

Figure 59 show the training accuracy in the CNN – LSTM with Stock Indicators. Accuracy on training data (orange line) stabilized near 100%. Accuracy on validation (blue) data was much more volatile. This may be a symptom of the training strategy used, where new data was added in batches, instead of providing the full dataset from the beginning. The accuracy on test data was 73%, the highest in all models. This may be due to coupling stock values with the natural language processing.

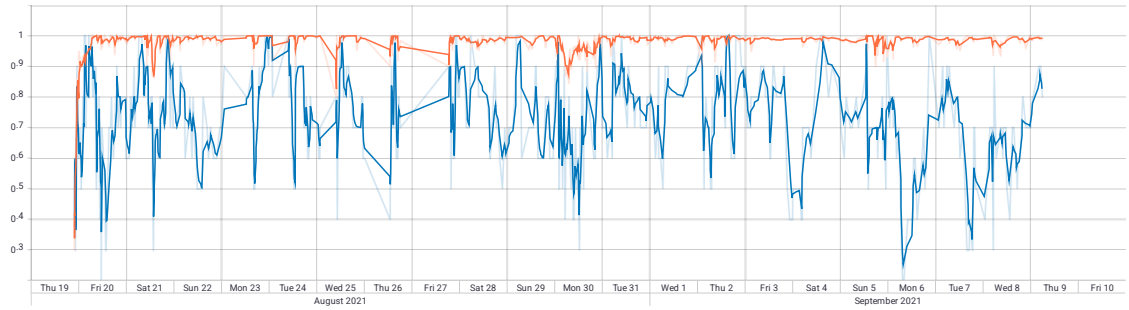


Figure 59: CNN - LSTM with Stock Indicators accuracy

The loss in Figure 60 seems to be a mirror image of the accuracy in Figure 59. The loss at the end was 13%.

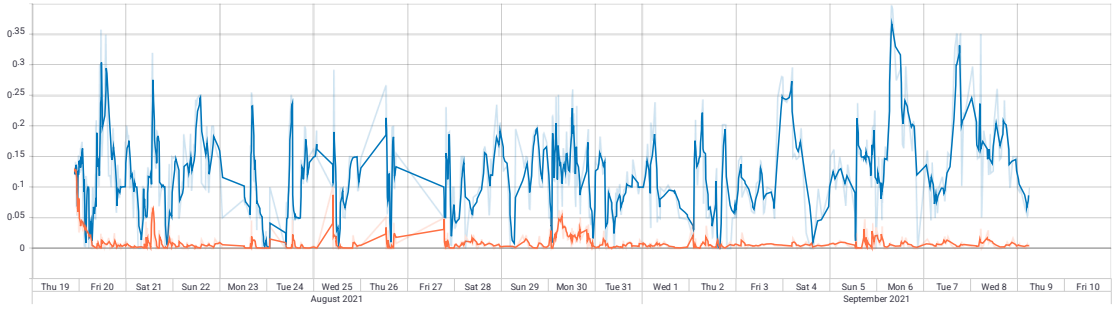


Figure 60: CNN - LSTM with Stock Indicators loss

In Figure 61, there is a description of the usage of GPU memory by the algorithm. It kept the heap usage below 1GB but ended having a higher fragmentation than other algorithms. This algorithm took the longest to train, reaching 20 days. This was also due to the strategy used as well as the complexity of the data structure which the algorithm used as input.

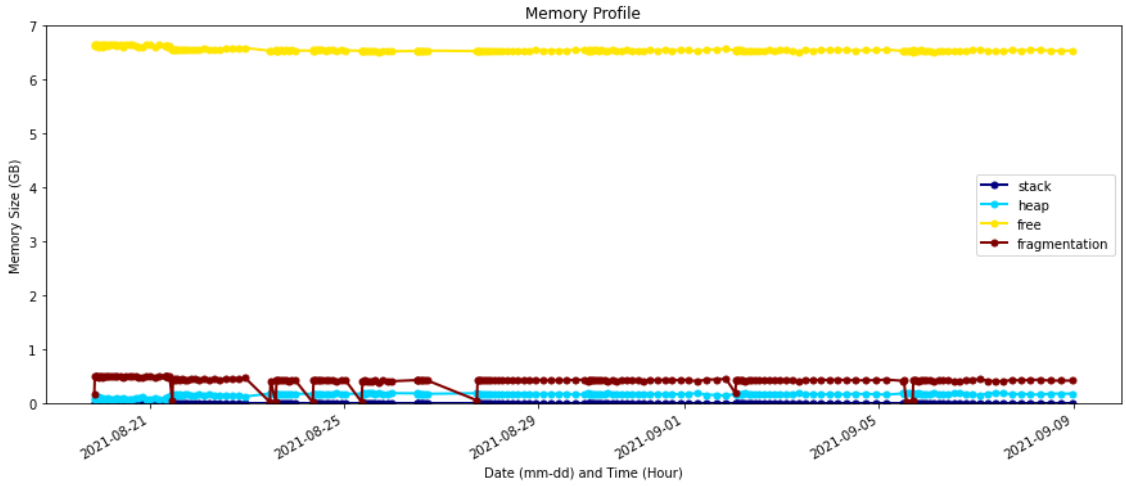


Figure 61: CNN - LSTM with Stock Indicators memory profile

Figure 62 shows the training accuracy on validation and training data for the CNN – GRU algorithm. It shows stabilization on train data (blue) and a considerable volatility in validation data (red). It finished with 63% accuracy on test data.

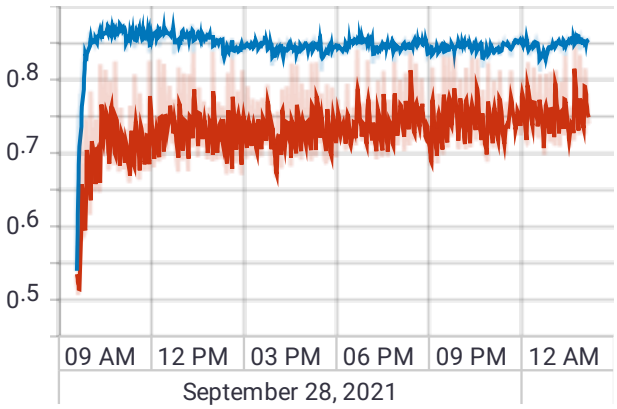


Figure 62: CNN - GRU train accuracy

Figure 63 shows loss on the CNN – GRU algorithm and follows the same colour code as Figure 62. Such as in the accuracy graph, the loss stabilizes on train data, but has volatile validation data. This could be a symptom of unrepresentative validation data. The final loss value was 11%.

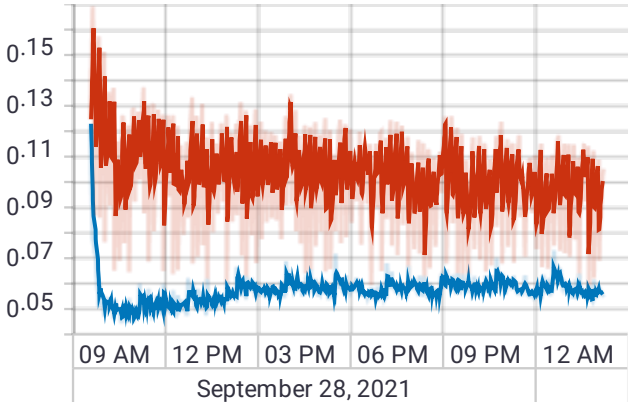


Figure 63: CNN – GRU train loss

Figure 64 shows a very little used heap, fragmentation, and stack GPU memory. It's possible that the profiler did not correctly register this data, or the algorithm uses very little GPU resources. It took 16 hours to train this model. The second-best time performance.

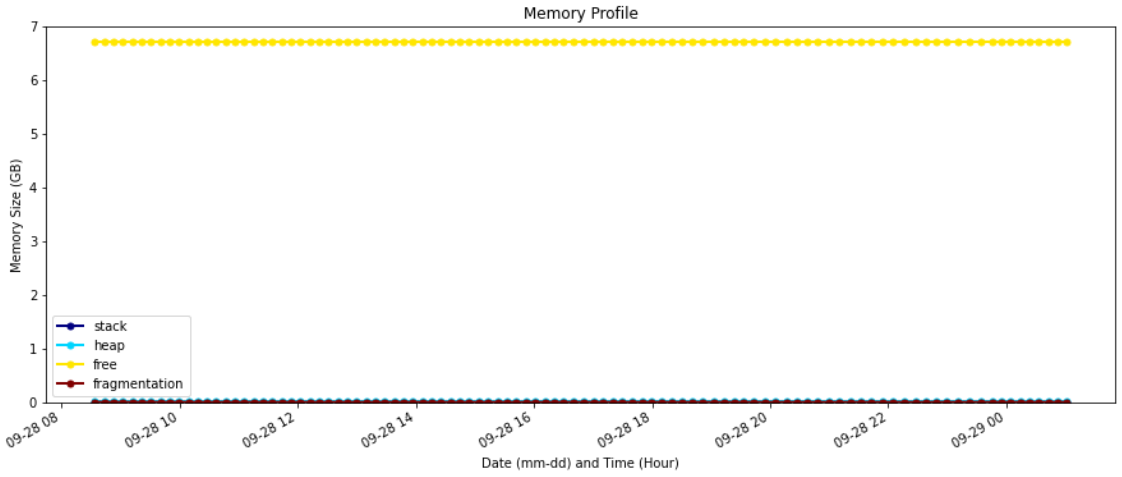


Figure 64: CNN - GRU memory profile

Figure 65 shows the accuracy on train data (orange) and validation data (blue). It seems to have a similar curve to the one in Figure 62. In the end, on test data, the accuracy result was 59%.

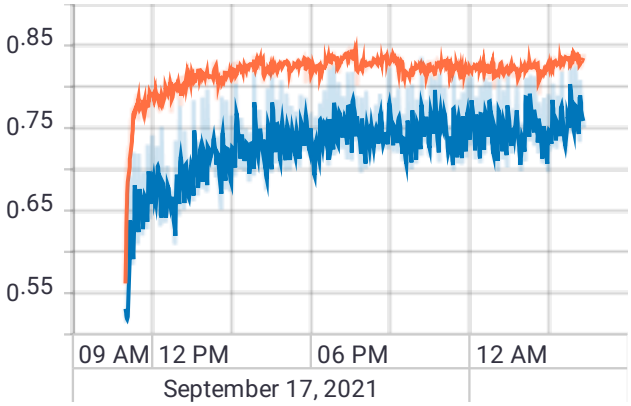


Figure 65: CNN - BiGRU train accuracy

Figure 66 shows the same pattern as Figure 65 does in relation to CNN – GRU, which may also indicate that validation data may be unbalanced for this algorithm. The loss was 9% in the end of the training phase.

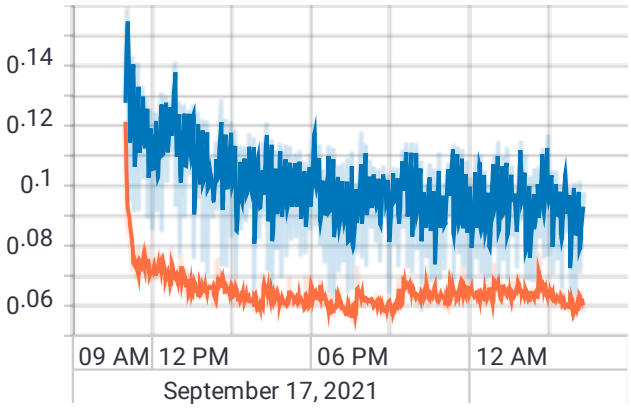


Figure 66: CNN - BiGRU train loss

Figure 67 shows a similar chart to Figure 64 which may be explained due to the similarities between both CNN – GRU and CNN – BiGRU. Timewise, this algorithm took 1 more hour to train than CNN – GRU, taking 17 hours in total.

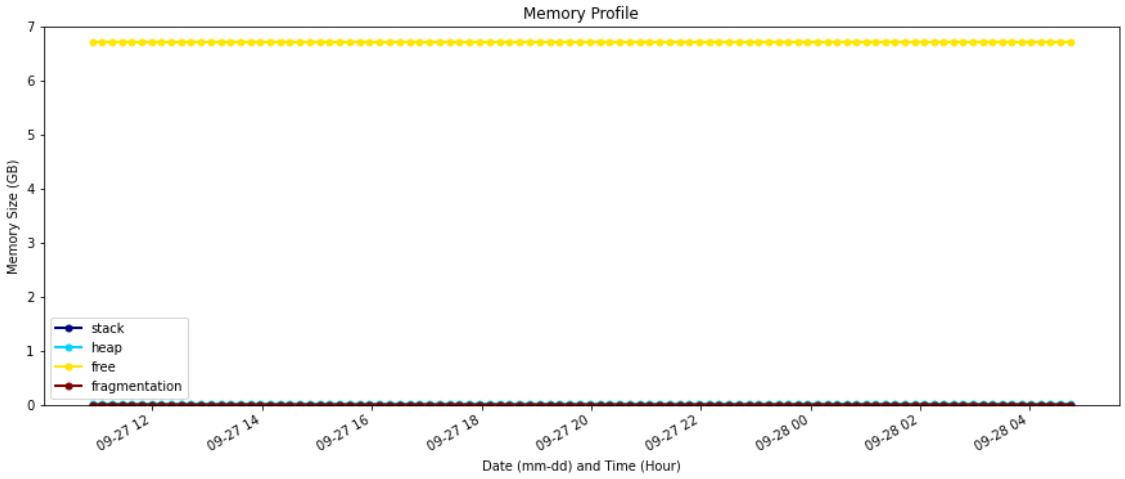


Figure 67: CNN-BiGRU train memory profile

In terms of accuracy, CNN – LSTM with Stock Indicators seems to stand out in relation to the other models, likely due to the mix of NLP and numerical stock value data. To validate the statistical significance in the difference of values, a hypothesis test can be used:

H_0: There is no significant difference between the accuracy of the CNN - LSTM with Stock Indicators and the other algorithms

H_1: There is significant difference between the accuracy of the CNN - LSTM with Stock Indicators and the other algorithms

Considering a significance of α = 5%, where

$$If\ \ p-value\ >\ \alpha, H_0\ is\ not\ rejected$$
$$If\ \ p-value\ \leq\ \alpha, H_0\ is\ rejected$$

$$\bar{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n} \approx\ 0.576; \tag{8}$$

$$S^2(n) = \frac{\sum_{i=1}^{n}[X_i - \bar{X}(n)]^2}{n-1} \rightarrow S^2(6)\ \approx\ 0.002; \tag{9}$$

$$t_n = \frac{[\bar{X}(n) - \mu]}{\sqrt{\frac{S^2(n)}{n}}}, where\ \mu\ =\ 0.726$$

$$t_7 = \frac{[\bar{X}(6) - \mu]}{\sqrt{\frac{S^2(6)}{6}}} = \frac{0.576 - 0.726}{\sqrt{\frac{0.002}{6}}} =\ -7.44$$

$$p - value \ = \ 0.003 \ and \ \alpha \ = \ 0.05$$

$$p - value \ \leq \ \alpha$$

(10)

We can reject the Hypothesis H_0.

What this tells us is that we cannot reject H_1, meaning there is the possibility that this algorithm has a significant difference of accuracy compared to the results from the other algorithms.

Another metric where CNN – LSTM with stock indicators stands out is the number of days it took to train it. To understand if this difference has statistical relevance, another hypothesis test can be used.

H_0: There is no significant difference between the time of training of the CNN - LSTM with Stock Indicators and the other algorithms

H_1: There is significant difference between the time of training of the CNN - LSTM with Stock Indicators and the other algorithms

Considering a significance of α = 5%, where

$$If \ \ p - value \ > \ \alpha, H_0 \ is \ not \ rejected$$
$$If \ \ p - value \ \leq \ \alpha, H_0 \ is \ rejected$$

$$\bar{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n} \approx \ 23;$$

(11)

$$S^2(n) = \frac{\sum_{i=1}^{n} [X_i - \bar{X}(n)]^2}{n-1} \ \rightarrow S^2(6) \ \approx \ 124;$$

(12)

$$t_n = \frac{[\bar{X}(n) - \ \mu]}{\sqrt{\frac{S^2(n)}{n}}}, where \ \mu \ = \ 480$$

$$t_7 = \frac{[\bar{X}(6) - \ \mu]}{\sqrt{\frac{S^2(6)}{6}}} = \frac{23 - 480}{\sqrt{\frac{124}{6}}} = \ -100.52$$

$$p - value \ < \ 0.00001 \ and \ \alpha \ = \ 0.05$$

$$p - value \ \leq \ \alpha$$

(13)

We can reject the Hypothesis H_0.

What we can take from these tests is that besides having a significantly higher accuracy, the CNN – LSTM with Stock Indicators algorithm also has a significantly higher training time.

Loss seems to be smaller than accuracy, meaning that possibly the rate of correct answers is higher than the rate of wrong ones. However, to confirm the difference is statistically significant we can validate with the t-test, using the following hypotheses:

H_0: There is no significant difference between the losses and the accuracy

H_1: There is significant difference between the losses and the accuracy

$$If\ \ p-value\ >\ \alpha, H_0\ is\ not\ rejected$$
$$If\ \ p-value\ \leq\ \alpha, H_0\ is\ rejected$$

$$\bar{X}_1(n) = \frac{\sum_{i=1}^{n} X_i}{n} \approx\ 0.597; \bar{X}_2(n) = \frac{\sum_{i=1}^{n} X_i}{n} \approx\ 0.111;$$

(14)

$$S^2(n) = \frac{\sum_{i=1}^{n}[(X_{i1}\ - X_{i2}) - \bar{X}(n)]^2}{n-1}\ \rightarrow S^2(6)\ \approx\ 0.009;$$

(15)

$$t_n = \frac{[\bar{X}_1(n) - \bar{X}_2(n)\ ]}{\sqrt{\frac{S^2(n)}{n}}}$$
$$t_n =\ 12.3$$
$$p-value\ =\ 0.000018\ and\ \alpha\ =\ 0.05$$
$$p-value\ \leq\ \alpha$$

(16)

We can reject the Hypothesis H_0.

This shows that the accuracy values are significantly higher than the loss values, showing a positive outcome concerning the number of correct predictions.

It seems that the CNN – LSTM with Stock Indicators is capable of a greater accuracy but at the same time takes much longer to train. This was confirmed by the different hypothesis tests. In terms of loss, it was not apparent at first, but CNN – LSTM (the one without Stock Index data) does show the worst results. This means it fails considerably more

predictions than the other algorithms. All the other algorithm's metrics seem to have low significant difference. The next section focuses on models which resulted from the training phase. The tests section is about exposing these models to new data. Data from different sources and with different contexts.

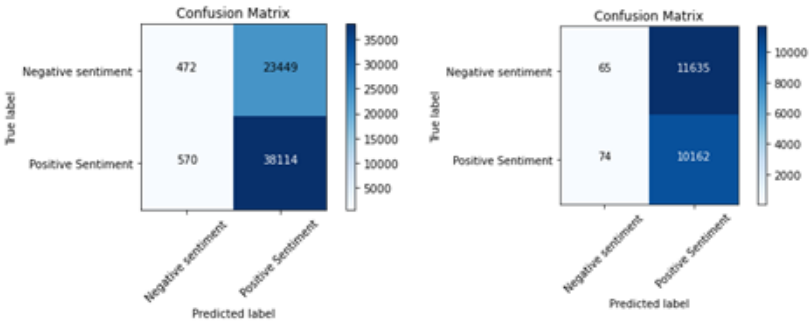
6.4.2 Tests

The test phase considers confusion matrices (Appendix B, Test Results) for each combination of dataset and model. Table 22 contains the average of each extracted indicator, obtained by averaging the results for the different datasets.

Table 22: Average test result for each model

Indicator	accuracy	precision	sensitivity	fallout	specificity	f1 score	MCC
CNN	0.575	0.577	0.99	0.989	0.011	0.727	-0.001
LSTM	0.507	0.573	0.6	0.604	0.396	0.579	-0.003
GRU	0.506	0.574	0.57	0.551	0.449	0.551	0.02
CNN - LSTM	0.525	0.596	0.601	0.591	0.409	0.597	-0.022
CNN - LSTM w/ Stock Indicators	0.486	0.489	0.316	0.324	0.676	0.366	0.03
CNN - GRU	0.521	0.579	0.704	0.698	0.302	0.625	0.007
CNN - BiGRU	0.518	0.574	0.665	0.669	0.331	0.611	-0.004

CNN has obtained the highest average accuracy result and CNN – LSTM w/Stock Indicators the lowest, the opposite of the results obtained in the training phase. A deep dive on the results obtained in the CNN, such as Figure 68, (Appendix B, Test Results) show that it is highly biased towards positive sentiment and thus the analysis of the results should not fall only on this indicator. The best accuracy results, discarding the biased CNN, come from the CNN – LSTM model.



CNN confusion matrix for all reddit data combined

CNN confusion matrix for AAPL data

Figure 68: 2 examples of CNN confusion matrices

Despite being the lowest average scorer, CNN – LSTM w/Stock Indicators had the best results when classifying the AAPL dataset, reaching a 69% accuracy (Appendix B, Table B 5), which can be a result of the use of numerical data for the stock indicators. It was also the model with the highest variance in the classification indicators. When using the MCC formula, it ends up as the one with the highest correlation coefficient while CNN ended up with the lowest, much as the training results. Using a 2-sample t-test it's possible to understand if there is significant difference between the average test scores in the MCC and accuracy metrics:

H_0: There is no significant difference between the average score of accuracy and MCC

H_1: There is a significant difference between the average score of accuracy and MCC

Considering a significance of α = 5%, where

$$If\ \ p-value\ >\ \alpha, H_0\ is\ not\ rejected$$
$$If\ \ p-value\ \leq\ \alpha, H_0\ is\ rejected$$

Before proceeding with the test, it's necessary to normalize the MCC metric (which varies between -1 and 1)

MCC = [0.4995, 0.4985,0.51,0.489,0.515,0.5035,0.498]

$$\bar{X}_1(n) = \frac{\sum_{i=1}^{n} X_i}{n} \approx\ 0.502; \bar{X}_2(n) = \frac{\sum_{i=1}^{n} X_i}{n} \approx\ 0.520;$$

(17)

$$S^2(n) = \frac{\sum_{i=1}^{n}[(X_{i1}\ - X_{i2}) - \bar{X}(n)]^2}{n-1}\ \rightarrow S^2(6)\ \approx\ 0.013;$$

(18)

$$t_n = \frac{[\bar{X}_1(n) - \bar{X}_2(n)\ ]}{\sqrt{\frac{S^2(n)}{n}}}$$
$$t_n =\ -3.41$$
$$p-value\ =\ 0.014\ and\ \alpha\ =\ 0.05$$
$$p-value\ >\ \alpha$$

(19)

We can reject the Hypothesis H_0.

This result shows that the mean values of MCC and accuracy are statistically different. This shows a disconnect between accuracy and MCC. To understand if all the metrics have the same statistical relevant difference in average samples, an ANOVA test can be used:

H_0: There is no significant difference between the average score of all metrics ($\mu_0 = \mu_1 = \mu_2 \ldots \mu_k$, where k is the number of independent comparison groups)

H_1: There is a significant difference between the average score of accuracy and MCC

Considering a significance of α = 5%, where

$$If\ p - value > \alpha, H_0\ is\ not\ rejected$$
$$If\ p - value \leq \alpha, H_0\ is\ rejected$$

```
library(readxl)
df <- read_excel("Livro1.xlsx")
View(df)

boxplot(df)

library(reshape2)
Mdados <- melt(df,variable.name = "metric" ,value.name = "value")

View(Mdados)
mod1 = lm(value ~ metric, data = Mdados)
anova(mod1)
```

Code 1: ANOVA test in R

Figure 69 shows the value distribution of the different metrics. The ANOVA test (Code 1) resulted in a p-value of 0.012, which is lower than 0.05. We can then reject the Hypothesis H_0. This means there is statistical difference between the mean values of the classification metrics.

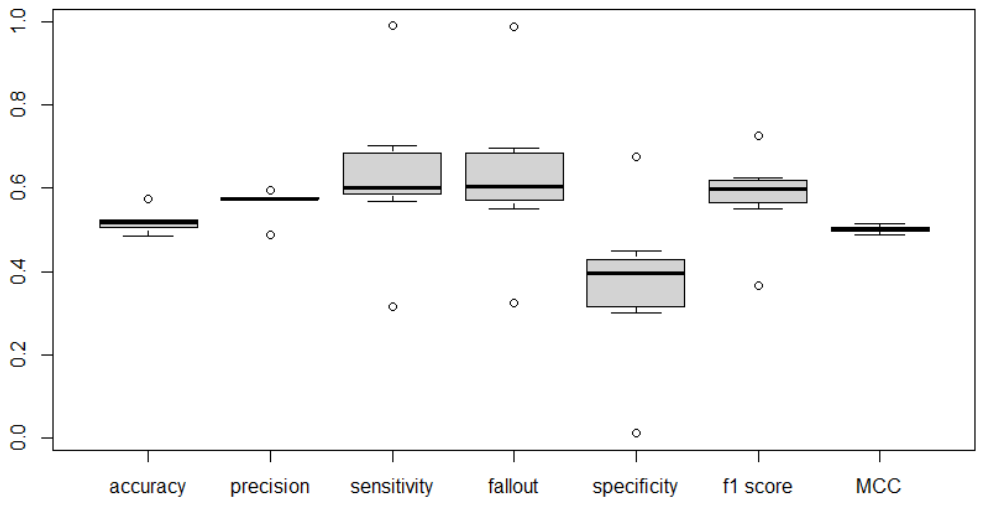


Figure 69: Test Metric distribution

Specificity is very low, possibly representing a low proportion of negatives that are correctly identified as such. It does have a big variance. This may have been a result of the bias to positive predictions in some models. This is supported by the results in both precision and sensitivity, the proportion of positive identifications which was correct and or the proportion of actual positives which was correctly identified. Fallout also shows a relation to this, since fallout is the ratio between the number of negative events wrongly categorized as positive (false positives) and the total number of actual negative events. Since there is a bias to positive sentiment, fallout tends to be higher, meaning there are more negatives which were classified as a positive. The F1 Score is intrinsically tied to the value of precision and sensitivity, since it is the harmonic mean of the 2 metrics.

Given the results which were obtained in the accuracy metric favour a biased model whereas MCC penalizes it (this is supported by the other metrics) the algorithm selected to run the investment simulation in the next section will be based on the MCC indicator. This indicator has a statistically significant difference to accuracy. CNN – LSTM with Stock performed the best on the MCC metric.

The selected datasets for the simulation will be the ones with the best performance for the selected model, which incidentally are the best results from all the samples. The first dataset is the AAPL stock data information, the dataset with the second best MCC and the best accuracy. The second dataset is the one with the best MCC result and second-best accuracy, the TSLA dataset. The next section will focus on simulating an investment strategy using the two mentioned dataset's results.

6.4.3 Real-world simulation

As stated before, to understand the value which comes from the trained models, this section covers the steps taken to simulate an investment strategy based on the model CNN – LSTM with Stock Indicators' prediction results on the AAPL dataset and the TSLA dataset.

The simulation will follow these rules:

- The starting balance is 1000$
- If the predicted sentiment is 1, the full balance is invested
- If the predicted sentiment is 0, the full balance is removed from the investment and placed into savings
- Each time the money is invested it will be completely, leaving savings empty
- In the end, the profit or loss is obtained by calculating the difference of balance and the initial investment
- The Return on investment (ROI) is the profit or loss divided by the initial investment
- To Provide a baseline, 1000$ are invested in the first day and never removed

Table 23 contains a summarized version of what happened in this simulation. The full detail is on Appendix B (Simulation Results).

Table 23: Summary of the simulation

Simulation	Baseline	Baseline ROI	Model Strategy	Model Strategy ROI
TSLA	1091.49$	9.1%	1043.95$	4.4%
AAPL	970.09$	-3.0%	979.04$	-2.1%

In both cases, the algorithm followed the trend for the financial index for the analysed period, between 13/09/2021 and 28/09/2021. The AAPL index fell 3%, while the TSLA increased 9.1% as shown by the simulation baseline. The AAPL ROI was higher than the baseline but still ended in a loss. Meanwhile the TSLA ROI was smaller than the baseline and ended in profit. These results seem to indicate that the model ended up smoothing the risk associated to the investment. These results show that there is some value to be extracted from the data. The use of numerical stock index data may have contributed to adding value to the sentiment data.

A different strategy may have had yielded different results. That is not however the scope of this thesis nor this simulation. The scope is to understand how Deep Learning is capable of extracting value from data. This simulation also does not take into consideration the cost of buying or selling stocks.

The next section takes all which was discussed previously and summarizes it, explaining possible outcomes in different scenarios, as well as possible future work.

7 Conclusion

Summarizing what was written in the previous chapters, Big Data has special characteristics which complicate its analysis, storage, management, etc. The defining attributes of Big Data (Volume, Veracity, Velocity, Variation and Value) come with problems associated to them. Due to these problems, different techniques have been employed to extract meaning from this type of data. Deep Learning is only one of the possible techniques.

Big Data technologies have been improving over time. It's possible to setup a cluster with relative ease. This project did not go deep into the capabilities of Hadoop and its related tools, opting to set up a 1-node DFS cluster as well as 1-node Spark cluster. This configuration allowed us to take advantage of Spark's optimized resource management for data processing.

This dissertation studies the use of Deep Learning to tackle Big Data problems, by using social network data. Financial market stock predictions were not the goal of this thesis. The goal was to use the predictions, and the steps which come before, such as training, data collection, etc. to understand how Deep Learning can be used with Big Data. Various datasets were used to train and test different models. Each dataset had sentence data which had to be prepared to be processed by Deep Learning algorithms. Due to the size of data, hardware optimizations, such as GPU parallelization, were used to avoid the Volume problems which affect other Traditional Machine Learning algorithms, which Deep Learning can take advantage of due to its multiprocessing capability.

Some Deep Learning algorithms require data structures which are too complex and too big to store in RAM. A possible way to avoid this problem is to break data in smaller batches and train the algorithms in incremental intervals. This was the technique used to train the most demanding algorithms featured in this dissertation. However, during processing, the Deep Learning Algorithms never used all the GPU RAM available, allowing them to be trained without any issue. Velocity is a point where Deep Learning algorithms may be useful if kept simple. In this dissertation we were able to train some models in just hours, and others had to be trained during weeks. The CNN – LSTM with Stock Indicators ended up being significantly worse in terms of training time than the others. This is a point which could influence the selection of the algorithm to use, as well as the method used to train it. In this case, the longest model was a very specific implementation which needed to be trained as if going through a time series of data.

Variety was not widely tested in this dissertation. The inputs of the different algorithms consisted of sentence and numerical data. Sentence data was converted to numbers before being fed to algorithms, allowing the inputs to be standardized. The algorithms then used embedding techniques to extract relation between the words in the phrases, assigning a weight to each expression towards the sentiment to stock data.

For the simulation phase, the model with the best MCC was chosen, given that the algorithm was the training sample maximiser for accuracy with a proved statistical significance. The MCC metric was selected over the accuracy in the test phase, as its average values were significantly different from the other metrics and the metric penalizes heavily bad predictions. Even though the model produced by the selected algorithm achieved the overall lowest accuracy score, it was still selected, as the model with the highest accuracy was biased on the positive sentiment, which lowered faith in the accuracy results.

In the Test and Simulation phases, the issues of Veracity and Value became apparent. Reddit and Twitter are not exactly moderated by specialists in the subject in analysis, and its data is mostly comprised of thoughts of people without much moderation. This means that the sentiment demonstrated by posts on those data sources may have no correlation to the stock values which were analysed. It does seem that the level of specialization of the users on Reddit is a bit higher than those on Twitter, due to better results being achieved on those datasets. Another explanation is that it could be related to the training data also originating on Reddit. The data seems to have some Value however, as it returned a profit in one of the simulations and smoothed the loss in the other. One thing the models were not able to do though was to break away from the market conditions, heading in the same way the market values did in that time interval. This can be attributed to the model producing a binary classification, which simply records when it is time to invest or to cash-out. A more complex classification scale or even a regression model could have produced different results.

This dissertation contributed to the understanding on the usage of the Deep Learning on unstructured data, as well on how to process Big Data efficiently using Deep Learning specialized hardware techniques. Another contribution was to understand how different techniques of Deep Neural Networks act on sentence data producing Sentiment. Also, how numerical data can enrich the techniques used on Sentiment data. For Stock Values this dissertation contributed on describing different methods of DNN and their outputs for this type of data. Finally, this dissertation contributed for Big Data management, processing, and extraction techniques.

For future work, it is possible that using sentiment from professional stock traders may help improve classification, as it is more plausible to be Valuable data with higher Veracity associated to it. This would benefit both the train and test phases. Another possible change would be to use different data contexts for training, as these models have only been trained on the context of DJIA stock data paired with Reddit news headlines. DJIA does not have big value changes, as it is an aggregation of stock data, so it's stock value usually increases, even though the dataset features the 2008 financial crash, the models ended up with an imbalanced dataset. Deep Learning models are usually capable of avoiding being biased in imbalanced datasets. This was not always the case as evidenced by the CNN model which favoured positive sentiment. In terms of Big Data technologies, future work could focus on the usage of multiple nodes for DFS as well as Spark, as well as other tools which were not used in this dissertation.

References

Abadi, D. J., Boncz, P. B. and Harizopoulos, S. (2009) 'Column-oriented database systems', *Proceedings of the VLDB Endowment*, 2(2), pp. 1664–1665. doi: 10.14778/1687553.1687625.

Abadi, M. *et al.* (2016) 'TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems'. Available at: http://arxiv.org/abs/1603.04467 (Accessed: 7 February 2021).

Afonso, L. M. *et al.* (2018) 'The case for API communicability evaluation: Introducing API-SI with examples from Keras', *arXiv*.

Agatonovic-Kustrin, S. and Beresford, R. (2000) 'Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research', *Journal of Pharmaceutical and Biomedical Analysis*. Elsevier, pp. 717–727. doi: 10.1016/S0731-7085(99)00272-1.

Al-Garadi, M. A. *et al.* (2019) 'Predicting Cyberbullying on Social Media in the Big Data Era Using Machine Learning Algorithms: Review of Literature and Open Challenges', *IEEE Access*, 7, pp. 70701–70718. doi: 10.1109/ACCESS.2019.2918354.

Albawi, S., Mohammed, T. A. and Al-Zawi, S. (2018) 'Understanding of a convolutional neural network', in *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*. Institute of Electrical and Electronics Engineers Inc., pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

Alguliyev, R. and Imamverdiyev, Y. (2014) 'Big Data: Big promises for information security', in *8th IEEE International Conference on Application of Information and Communication Technologies, AICT 2014 - Conference Proceedings*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICAICT.2014.7035946.

Ali, S. M. *et al.* (2016) 'Big Data visualization: Tools and challenges', in *Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016*. Institute of Electrical and Electronics Engineers Inc., pp. 656–660. doi: 10.1109/IC3I.2016.7918044.

Alwidian, J. *et al.* (2020) 'Big Data Ingestion and Preparation Tools', *Modern Applied Science*, 14(9). doi: 10.5539/mas.v14n9p12.

Anikin, D., Borisenko, O. and Nedumov, Y. (2019) 'Labeled Property Graphs: SQL or NoSQL?', in *Proceedings - 2019 Ivannikov Memorial Workshop, IVMEM 2019*. Institute of Electrical and Electronics Engineers Inc., pp. 7–13. doi: 10.1109/IVMEM.2019.00007.

Bahrampour, S. *et al.* (2015) 'COMPARATIVE STUDY OF CAFFE, NEON, THEANO, AND TORCH FOR DEEP LEARNING', *ArXiv*, 2(1), pp. 1–14. Available at: http://deeplearning4j.org/accuracy.html. (Accessed: 11 February 2021).

Bandi, R. and Anitha, G. (2018) 'Machine learning based oozie workflow for hive query schedule mechanism', in *Proceedings of the International Conference on Smart Systems and Inventive Technology, ICSSIT 2018*. Institute of Electrical and Electronics Engineers Inc., pp. 513–517. doi: 10.1109/ICSSIT.2018.8748711.

Basiri, M. E. *et al.* (2021) 'ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis', *Future Generation Computer Systems*, 115, pp. 279–294. doi: 10.1016/j.future.2020.08.005.

Bengio, Y. (2009) 'Learning deep architectures for AI', *Foundations and Trends in Machine Learning*, 2(1), pp. 1–27. doi: 10.1561/2200000006.

Bollen, J., Mao, H. and Zeng, X. (2011) 'Twitter mood predicts the stock market', *Journal of Computational Science*, 2(1), pp. 1–8. doi: 10.1016/j.jocs.2010.12.007.

Bonchi, F. and Ferrari, E. (2010) *Privacy-aware knowledge discovery: Novel applications and new techniques*, *Privacy-Aware Knowledge Discovery: Novel Applications and New Techniques*. doi: 10.1201/b10373.

Bordes, A. *et al.* (2011) 'Learning structured embeddings of knowledge bases', in *Proceedings of the National Conference on Artificial Intelligence*, pp. 301–306. Available at: www.aaai.org (Accessed: 20 December 2020).

Buchanan, J. and Gardiner, L. (2003) 'A comparison of two reference point methods in multiple objective mathematical programming', *European Journal of Operational Research*, 149(1), pp. 17–34. doi: 10.1016/S0377-2217(02)00487-3.

Bürkner, P. C., Gabry, J. and Vehtari, A. (2020) 'Approximate leave-future-out cross-validation for Bayesian time series models', *Journal of Statistical Computation and Simulation*, pp. 2499–2523. doi: 10.1080/00949655.2020.1783262.

Can, U. and Alatas, B. (2017) 'Big social network data and sustainable economic development', *Sustainability (Switzerland)*, 9(11). doi: 10.3390/su9112027.

Cataldi, M., Di Caro, L. and Schifanella, C. (2010) 'Emerging topic detection on Twitter based on temporal and social terms evaluation', in *Proceedings of the 10th International Workshop on Multimedia Data Mining, MDMKDD '10*. New York, New York, USA: ACM Press, pp. 1–10. doi: 10.1145/1814245.1814249.

Cavanillas, J. M., Curry, E. and Wahlster, W. (2016) *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*, *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*. doi: 10.1007/978-3-319-21569-3.

Chatterjee, A. *et al.* (2019) 'Understanding Emotions in Text Using Deep Learning and Big Data', *Computers in Human Behavior*, 93, pp. 309–317. doi: 10.1016/j.chb.2018.12.029.

Chen, G. (2016) 'A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation'. Available at: http://arxiv.org/abs/1610.02583 (Accessed: 13 February 2021).

Chen, M. *et al.* (2014) *Big Data*. Cham: Springer International Publishing (SpringerBriefs in Computer Science). doi: 10.1007/978-3-319-06245-7.

Cho, Y., wang, J. and Lee, D. (2012) 'Identification of effective opinion leaders in the diffusion of technological innovation: A social network approach', *Technological Forecasting and Social Change*, 79(1), pp. 97–106. doi: 10.1016/j.techfore.2011.06.003.

Chung, J. *et al.* (2014) 'Empirical Evaluation of Gated Recurrent Neural Networks on Sequence

Modeling'. Available at: https://arxiv.org/abs/1412.3555v1 (Accessed: 10 October 2021).

Cleary, M. (2019) *Python Data Science Handbook*, *Journal of Chemical Information and Modeling*.

Condie, T. *et al.* (2010) 'MapReduce online', in *Proceedings of NSDI 2010: 7th USENIX Symposium on Networked Systems Design and Implementation*, pp. 313–327.

Dahiya, V. (2020) 'Parallel Approaches of Utility Mining for Big Data Sandeep Dalal', 17(2). doi: 10.14704/WEB/V17I2/WEB17014.

Ding, X. *et al.* (2015) 'Deep Learning for event-driven stock prediction', in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2327–2333.

Doleck, T. *et al.* (2020) 'Predictive analytics in education: a comparison of Deep Learning frameworks', *Education and Information Technologies*, 25(3), pp. 1951–1963. doi: 10.1007/s10639-019-10068-4.

Ekka, S. and Jayapandian, N. (2020) 'Big Data Analytics Tools and Applications for Modern Business World', in *Proceedings of the International Conference on Electronics and Sustainable Communication Systems, ICESC 2020*. Institute of Electrical and Electronics Engineers Inc., pp. 587–592. doi: 10.1109/ICESC48915.2020.9155704.

Ferrari, A. and Russo, M. (2016) *Introducing Microsoft Power BI - Alberto Ferrari, Marco Russo - Google Livros*. Available at: https://books.google.pt/books?hl=pt-PT&lr=&id=U1qsDAAAQBAJ&oi=fnd&pg=PT8&dq=power+bi&ots=pAMYVZlZSA&sig=bnQS3nasGcdQUCrccCocPhFMK4A&redir_esc=y#v=onepage&q=power bi&f=false (Accessed: 7 February 2021).

Gao, Z. and Wang, X. (2019) 'Deep Learning', *EEG Signal Processing and Feature Extraction*, pp. 325–333. doi: 10.1007/978-981-13-9113-2_16.

Goldsborough, P. (2016) 'A Tour of TensorFlow'. Available at: https://github.com/soumith/cudnn.torch (Accessed: 10 February 2021).

Grattarola, D. and Alippi, C. (2021) 'Graph Neural Networks in TensorFlow and Keras with Spektral [Application Notes]', *IEEE Computational Intelligence Magazine*, 16(1), pp. 99–106. doi: 10.1109/MCI.2020.3039072.

Group, W. M. (2013) 'QFD: Product Excellence using Six Sigma: QFD', pp. 1–61.

Gudivada, V. N. *et al.* (2015) 'GUEST EDITORS' INTRODUCTION Big Data: Promises and Problems', *Computer (2015)*, p. 21. Available at: http://www.cs.rug.nl/~roe/courses/isc/BigDataPromises.pdf (Accessed: 23 February 2021).

Gudivada, V. N., Rao, D. and Raghavan, V. V. (2014) 'NoSQL Systems for Big Data Management', in. Institute of Electrical and Electronics Engineers (IEEE), pp. 190–197. doi: 10.1109/services.2014.42.

Gui, H. (2019) *STOCK PREDICTION BASED ON SOCIAL MEDIA DATA VIA SENTIMENT ANALYSIS A study on Reddit*. Available at: https://trepo.tuni.fi/handle/10024/118242 (Accessed: 27 December 2020).

Haerder, T. and Reuter, A. (1983) 'Principles of transaction-oriented database recovery', *ACM Computing Surveys (CSUR)*, 15(4), pp. 287–317. doi: 10.1145/289.291.

Handelman, G. S. *et al.* (2019) 'Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods', *American Journal of Roentgenology*, 212(1), pp. 38–43. doi: 10.2214/AJR.18.20224.

Hassan, A. and Mahmood, A. (2018) 'Convolutional Recurrent Deep Learning Model for Sentence Classification', *IEEE Access*, 6, pp. 13949–13957. doi: 10.1109/ACCESS.2018.2814818.

Hunt, P. *et al.* (2019) 'ZooKeeper: Wait-free coordination for internet-scale systems', in *Proceedings of the 2010 USENIX Annual Technical Conference, USENIX ATC 2010*.

Ide, H. and Kurita, T. (2017) 'Improvement of learning for CNN with ReLU activation by sparse regularization', *Proceedings of the International Joint Conference on Neural Networks*, 2017-May, pp. 2684–2691. doi: 10.1109/IJCNN.2017.7966185.

Prabha, M. and Srikanth, G. (2019) 'Survey of Sentiment Analysis Using Deep Learning Techniques', in *Proceedings of 1st International Conference on Innovations in Information and Communication Technology, ICIICT 2019*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICIICT1.2019.8741438.

Inoubli, W. *et al.* (2018) 'An experimental survey on Big Data frameworks', *Future Generation Computer Systems*, 86, pp. 546–564. doi: 10.1016/j.future.2018.04.032.

Irsoy, O. and Cardie, C. (2014) 'Deep recursive neural networks for compositionality in language', in *Advances in Neural Information Processing Systems*, pp. 2096–2104.

Jenkins, B. (2016) *BIG DATA ANALYTICS USING HADOOP TOOLS*.

Jeong, Y. S., Hassan, H. and Sangaiah, A. K. (2019) 'Machine learning on Big Data for future computing', *Journal of Supercomputing*. Springer New York LLC, pp. 2925–2929. doi: 10.1007/s11227-019-02872-z.

Ji, C. *et al.* (2012) 'Big Data processing in cloud computing environments', in *Proceedings of the 2012 International Symposium on Pervasive Systems, Algorithms, and Networks, I-SPAN 2012*, pp. 17–23. doi: 10.1109/I-SPAN.2012.9.

Jia, Y. *et al.* (2014) 'Caffe: Convolutional architecture for fast feature embedding', in *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, pp. 675–678. doi: 10.1145/2647868.2654889.

Johnson, N. S., Vulimiri, P. S. and Zhang, X. (2020) 'Machine Learning for Materials Developments in Metals Additive Manufacturing. (arXiv:2005.05235v1 [physics.app-ph])'. Available at: https://www.researchgate.net/publication/341310767_Machine_Learning_for_Materials_Developments_in_Metals_Additive_Manufacturing (Accessed: 6 September 2021).

Karaivanov, A. (2020) 'A social network model of COVID-19', *PLoS ONE*, 15(10 October), p. e0240878. doi: 10.1371/journal.pone.0240878.

Ketkar, N. (2017) *Deep Learning with Python*, *Deep Learning with Python*. Apress. doi: 10.1007/978-1-4842-2766-4.

Khan, N. *et al.* (2014) 'Big Data: Survey, technologies, opportunities, and challenges', *Scientific World Journal*. Hindawi Publishing Corporation. doi: 10.1155/2014/712826.

Khanaferov, D., Luc, C. and Wang, T. (2014) 'Social network data mining using natural language processing and density based clustering', in *Proceedings - 2014 IEEE International Conference on Semantic Computing, ICSC 2014*. IEEE Computer Society, pp. 250–251. doi: 10.1109/ICSC.2014.48.

Kumar Vavilapalli, V. *et al.* (2013) 'Apache Hadoop YARN: Yet Another Resource Negotiator', 13, pp. 1–3. doi: 10.1145/2523616.2523633.

L'Heureux, A. *et al.* (2017) 'Machine Learning with Big Data: Challenges and Approaches', *IEEE Access*, 5, pp. 7776–7797. doi: 10.1109/ACCESS.2017.2696365.

Lago, J., De Ridder, F. and De Schutter, B. (2018) 'Forecasting spot electricity prices: Deep Learning approaches and empirical comparison of traditional algorithms', *Applied Energy*, 221, pp. 386–405. doi: 10.1016/j.apenergy.2018.02.069.

Lecun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep Learning', *Nature*. Nature Publishing Group, pp. 436–444. doi: 10.1038/nature14539.

Lee, S., Min, S. J. and Eigenmann, R. (2009) 'Open MP to GPGPU: A compiler framework for automatic translation and optimization', *ACM SIGPLAN Notices*, 44(4), pp. 101–110. doi: 10.1145/1594835.1504194.

Liu, G. and Guo, J. (2019) 'Bidirectional LSTM with attention mechanism and convolutional layer for text classification', *Neurocomputing*, 337, pp. 325–338. doi: 10.1016/j.neucom.2019.01.078.

Liu, W. *et al.* (2017) 'A survey of deep neural network architectures and their applications', *Neurocomputing*, 234, pp. 11–26. doi: 10.1016/j.neucom.2016.12.038.

Londhe, A. and Prasada Rao, P. V. R. D. (2018) 'Platforms for Big Data analytics: Trend towards hybrid era', in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017*. Institute of Electrical and Electronics Engineers Inc., pp. 3235–3238. doi: 10.1109/ICECDS.2017.8390056.

Lourenço, J. R. *et al.* (2015) 'NoSQL Databases: A Software Engineering Perspective', in *Rocha A., Correia A., Costanzo S., Reis L. (eds) New Contributions in Information Systems and Technologies. Advances in Intelligent Systems and Computing, vol 353. Springer, Cham.*, pp. 741–750. doi: 10.1007/978-3-319-16486-1_73.

Lovalekar, S. (2014) 'Big Data : An Emerging Trend In Future', *International Journal of Computer Science and Information Technologies,* 5(1), pp. 538–541. Available at: www.ijcsit.com (Accessed: 8 February 2021).

Madden, S. (2012) 'From databases to Big Data', *IEEE Internet Computing*, pp. 4–6. doi: 10.1109/MIC.2012.50.

Maia, M. *et al.* (2018) 'WWW'18 Open Challenge: Financial Opinion Mining and Question Answering', *Www*, 1, pp. 1941–1942. doi: 10.1145/3184558.

Mashtizadeh, A. J. *et al.* (2013) 'Replication, history, and grafting in the Ori file system', in

SOSP 2013 - Proceedings of the 24th ACM Symposium on Operating Systems Principles, pp. 151–166. doi: 10.1145/2517349.2522721.

Meng, J. *et al.* (2012) 'DGraph: Algorithms for shortgun reads assembly using De Bruijn graph', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, Berlin, Heidelberg, pp. 14–21. doi: 10.1007/978-3-642-35606-3_2.

Muzio, T. Di (2021) *GameStop Capitalism: Wall Street vs. The Reddit Rally (Part 1) – Capital As Power*. Available at: http://bnarchives.yorku.ca/673/3/20210200_di_muzio_gamestop_capitalism_part_1.html (Accessed: 14 February 2021).

Nambiar, R. *et al.* (2013) 'A look at challenges and opportunities of Big Data analytics in healthcare', in *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*, pp. 17–22. doi: 10.1109/BigData.2013.6691753.

Nguyen, H. and Nguyen, M. Le (2018) 'A Deep Neural Architecture for Sentence-Level Sentiment Classification in Twitter Social Networking', in *Communications in Computer and Information Science*. Springer Verlag, pp. 15–27. doi: 10.1007/978-981-10-8438-6_2.

Nwagbara, U. (2013) 'The Effects of Social Media on Environmental Sustainability Activities of Oil and Gas Multinationals in Nigeria', *Thunderbird International Business Review*, 55(6), pp. 689–697. doi: 10.1002/tie.21584.

Oncharoen, P. and Vateekul, P. (2018) 'Deep Learning for Stock Market Prediction Using Event Embedding and Technical Indicators', in *ICAICTA 2018 - 5th International Conference on Advanced Informatics: Concepts Theory and Applications*. Institute of Electrical and Electronics Engineers Inc., pp. 19–24. doi: 10.1109/ICAICTA.2018.8541310.

Orduna-Malea, E. and Lopez-Cozar, E. D. (2018) 'Dimensions: re-discovering the ecosystem of scientific information', *Profesional de la Informacion*, 27(2), pp. 420–431. doi: 10.3145/epi.2018.mar.21.

Otter, D. W., Medina, J. R. and Kalita, J. K. (2018) 'A survey of the usages of Deep Learning in natural language processing', *arXiv*. arXiv. doi: 10.1109/tnnls.2020.2979670.

Oussous, A. *et al.* (2018) 'Big Data technologies: A survey', *Journal of King Saud University - Computer and Information Sciences*. King Saud bin Abdulaziz University, pp. 431–448. doi: 10.1016/j.jksuci.2017.06.001.

Owen, S. *et al.* (2011) *Mahout in Action*, *Online*. Available at: http://www.manning.com/owen/ (Accessed: 7 February 2021).

Pang, L. and Liu, Y. (2020) 'Construction and application of a financial Big Data analysis model based on machine learning', *Revue d'Intelligence Artificielle*, 34(3), pp. 345–350. doi: 10.18280/ria.340313.

Parvandeh, S. *et al.* (2020) 'Consensus features nested cross-validation', *Bioinformatics*, 36(10), pp. 3093–3098. doi: 10.1093/BIOINFORMATICS/BTAA046.

Paszke, A. *et al.* (2017a) *Automatic differentiation in pytorch*. Available at: https://openreview.net/forum?id=BJJsrmfCZ (Accessed: 11 February 2021).

Paszke, A. *et al.* (2017b) *Automatic differentiation in pytorch*. Available at: https://openreview.net/forum?id=BJJsrmfCZ (Accessed: 10 February 2021).

Pavlo, A. and Aslett, M. (2016) *What's Really New with NewSQL?*

Pearson, K. (1895) 'VII. Note on regression and inheritance in the case of two parents', *Proceedings of the Royal Society of London*, 58(347–352), pp. 240–242. doi: 10.1098/rspl.1895.0041.

Petrovic, O. and Kittl, C. (2003) 'Capturing the value proposition of a product or service'.

Prasad, N. R., Almanza-Garcia, S. and Lu, T. T. (2009) 'Anomaly detection', *Computers, Materials and Continua*, 14(1), pp. 1–22. doi: 10.1145/1541880.1541882.

Rains, J. A. (2009) 'What are the functions of function analysis', *49th Annual Conference of SAVE International 2009*, pp. 225–232.

Rajaraman, V. (2016) 'Big Data analytics', *Resonance*, 21(8), pp. 695–716. doi: 10.1007/s12045-016-0376-7.

Ranjan, R. (2014) 'Streaming Big Data Processing in Datacenter Clouds', *IEEE Cloud Computing*, 1(1), pp. 78–83. doi: 10.1109/MCC.2014.22.

Rezaeinia, S. M. *et al.* (2019) 'Sentiment analysis based on improved pre-trained word embeddings', *Expert Systems with Applications*, 117, pp. 139–147. doi: 10.1016/j.eswa.2018.08.044.

Rodrigues, M., Santos, M. Y. and Bernardino, J. (2017) 'Experimental evaluation of Big Data analytical tools', in *Lecture Notes in Business Information Processing*, pp. 121–127. doi: 10.1007/978-3-030-11395-7_12.

Roh, Y., Heo, G. and Whang, S. E. (2018) 'A survey on data collection for machine learning: A Big Data - AI integration perspective', *arXiv*. doi: 10.1109/tkde.2019.2946162.

Rossi, R. A. and Ahmed, N. K. (2015) 'Role Discovery in Networks', *IEEE Transactions on Knowledge and Data Engineering*, 27(4), pp. 1112–1131. doi: 10.1109/TKDE.2014.2349913.

Ruggieri, S., Pedreschi, D. and Turini, F. (2010) 'Data mining for discrimination discovery', *ACM Transactions on Knowledge Discovery from Data*, 4(2), pp. 1–40. doi: 10.1145/1754428.1754432.

Russom, P. and Org, T. (2011) *BIG DATA ANALYTICS FOURTH QUARTER 2011 TDWI RE SE A RCH Co-sponsored by BIG DATA A N A LY TIC S FOURTH QUARTER 2011 TDWI BEST PRACTICES REPORT Introduction to Big Data Analytics*.

Saaty, R. W. (1987) 'The analytic hierarchy process-what it is and how it is used', *Mathematical Modelling*, 9(3–5), pp. 161–176. doi: 10.1016/0270-0255(87)90473-8.

Samek, W. *et al.* (2021) 'Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications', *Proceedings of the IEEE*, 109(3), pp. 247–278. doi: 10.1109/JPROC.2021.3060483.

Sammer, E. (2012) *Hadoop Operations*. Available at: https://books.google.pt/books?hl=pt-PT&lr=&id=W5VWrrCOuQ8C&oi=fnd&pg=PR5&ots=PF9mOxcR2-

&sig=VBIiMM03R9_hEYnbzv9y9oCN_70&redir_esc=y#v=onepage&q&f=false (Accessed: 8 February 2021).

Satyanarayanan, M. *et al.* (1990) *Coda: A Highly Available File System for a Distributed Workstation Environment*, *IEEE TRANSACTIONS ON COMPUTERS*.

Schmid, H. (1994) 'Part-of-Speech Tagging with Neural Networks'. Available at: http://arxiv.org/abs/cmp-lg/9410018 (Accessed: 27 February 2021).

Šestak, M. *et al.* (2021) 'Applying k-vertex cardinality constraints on a Neo4j graph database', *Future Generation Computer Systems*, 115, pp. 459–474. doi: 10.1016/j.future.2020.09.036.

Shi, T. and Liu, Z. (2014) 'Linking GloVe with word2vec'. Available at: http://arxiv.org/abs/1411.5595 (Accessed: 27 February 2021).

Singh, P. and Singh, P. (2019) 'Airflow', in *Learn PySpark*. Apress, pp. 67–84. doi: 10.1007/978-1-4842-4961-1_4.

Sohangir, S. *et al.* (2018) 'Big Data: Deep Learning for financial sentiment analysis', *Journal of Big Data*, 5(1). doi: 10.1186/s40537-017-0111-6.

Stimmel, C. L. (2016) *Big Data analytics strategies for the smart grid*, *Big Data Analytics Strategies for the Smart Grid*. doi: 10.1201/b17228.

Tan, W. *et al.* (2013) 'Social-network-sourced Big Data analytics', *IEEE Internet Computing*, 17(5), pp. 62–69. doi: 10.1109/MIC.2013.100.

Tereso, M. and Bernardino, J. (2011) 'Open source business intelligence tools for SMEs', *6th Iberian Conference on Information Systems and Technologies*, CISTI 2011, pp. 1–4.

The Theano Development Team *et al.* (2016) 'Theano: A Python framework for fast computation of mathematical expressions'. Available at: https://groups.google.com/group/theano-dev/ (Accessed: 10 February 2021).

Tokui, S. *et al.* (2015) 'Chainer: a Next-Generation Open Source Framework for Deep Learning'.

Verma, A., Mansuri, A. H. and Jain, N. (2016) 'Big Data management processing with Hadoop MapReduce and spark technology: A comparison', in *2016 Symposium on Colossal Data Analysis and Networking, CDAN 2016*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/CDAN.2016.7570891.

Vukotic, A. *et al.* (2015) *Neo4j in action*. Available at: www.manning.com. (Accessed: 31 January 2021).

Wadkar, S. *et al.* (2014) 'Apache Ambari', in *Pro Apache Hadoop*. Apress, pp. 399–401. doi: 10.1007/978-1-4302-4864-4_20.

Wang, W. Y. C. and Wang, Y. (2020) 'Analytics in the era of Big Data: The digital transformations and value creation in industrial marketing', *Industrial Marketing Management*, 86, pp. 12–15. doi: 10.1016/J.INDMARMAN.2020.01.005.

Wen, C. and Yang, X. (2020) 'Visualizing Train Delays Using Tableau and the Framework of a Delay Impact Visualization System', *Journal of Big Data Analytics in Transportation*, 2(2), pp. 75–91. doi: 10.1007/s42421-020-00018-9.

Wen, S. and Li, J. (2018) 'Recurrent convolutional neural network with attention for twitter and yelp sentiment classification arc model for sentiment classification', in *ACM International Conference Proceeding Series*. New York, NY, USA: Association for Computing Machinery, pp. 1–7. doi: 10.1145/3302425.3302468.

Wirth, R. (2000) 'CRISP-DM : Towards a Standard Process Model for Data Mining', *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, (24959), pp. 29–39.

Xie, J. *et al.* (2018) 'A Survey on Machine Learning-Based Mobile Big Data Analysis: Challenges and Applications', *Wireless Communications and Mobile Computing*. Hindawi Limited. doi: 10.1155/2018/8738613.

Yang, Z. *et al.* (2016) *Hierarchical Attention Networks for Document Classification*.

Yin, W. *et al.* (2017) 'Comparative Study of CNN and RNN for Natural Language Processing'. Available at: http://arxiv.org/abs/1702.01923 (Accessed: 27 February 2021).

Zhang, Q. *et al.* (2015) 'Automatic detection of rumor on social network', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 113–122. doi: 10.1007/978-3-319-25207-0_10.

Zhou, L. *et al.* (2017) 'Machine learning on Big Data: Opportunities and challenges', *Neurocomputing*, 237, pp. 350–361. doi: 10.1016/j.neucom.2017.01.026.

Zhou, P. *et al.* (2018) 'Tensor factorization for low-rank tensor completion', *IEEE Transactions on Image Processing*, 27(3), pp. 1152–1163. doi: 10.1109/TIP.2017.2762595.

Zhou, X. and Chen, L. (2014) 'Event detection over twitter social media streams', *VLDB Journal*, 23(3), pp. 381–400. doi: 10.1007/s00778-013-0320-3.

Zhou, X., Wan, X. and Xiao, J. (2016) 'Attention-based LSTM network for cross-lingual sentiment classification', in *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 247–256. doi: 10.18653/v1/d16-1024.

Appendix A

AHP calculations

(1) RC

$$Ax = \lambda_{max} x,$$

$$A = \begin{bmatrix} 1 & 2 & 3 & 2 \\ 1/2 & 1 & 2 & 1/2 \\ 1/3 & 1/2 & 1 & 1/3 \\ 1/2 & 2 & 3 & 1 \end{bmatrix}, x = \begin{bmatrix} 0.41 \\ 0.19 \\ 0.11 \\ 0.29 \end{bmatrix}$$

Replacing the variables by the matrixes we get:

$$\begin{bmatrix} 1 & 2 & 3 & 2 \\ 1/2 & 1 & 2 & 1/2 \\ 1/3 & 1/2 & 1 & 1/3 \\ 1/2 & 2 & 3 & 1 \end{bmatrix} * \begin{bmatrix} 0.41 \\ 0.19 \\ 0.11 \\ 0.29 \end{bmatrix} = \lambda_{max} \begin{bmatrix} 0.41 \\ 0.19 \\ 0.11 \\ 0.29 \end{bmatrix}$$

Solving for λ_{max}:

$$\lambda_{max} = \begin{bmatrix} 1.70 \\ 0.76 \\ 0.44 \\ 1.20 \end{bmatrix} \div \begin{bmatrix} 0.41 \\ 0.19 \\ 0.11 \\ 0.29 \end{bmatrix}$$

Division of two matrixes:

$$\lambda_{max} = average(1.70/0.41, 0.76/0.19, 0.44/0.11, 1.20/0.29),$$
$$\lambda_{max} = 4.07$$

Obtaining IC:

$$IC = \frac{\lambda_{max} - n}{n-1}, n = 4$$
$$IC = \frac{4{,}07 - 4}{4-1} = 0.024$$

RC is then obtained via the equation:

$$RC = \frac{IC}{IR}, where\ IR\ is\ an\ index\ of\ \text{Table 14}$$

$$RC = \frac{0.024}{0.9} = 0.026$$

(2) Alternatives Relative Priorities

(a) Accuracy

Table A 1: Accuracy AHP comparison matrix with column sum

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM
CNN	1	3	2	2	1/4
RNN	1/3	1	1/2	1/3	1/6
LSTM	1/2	2	1	1/2	1/5
CNN+RNN	1/2	3	2	1	1/3
CNN+LSTM	4	6	5	3	1
Sum	6.33	15.00	10.50	6.83	1.95

Table A 2: Normalized accuracy comparison matrix with relative priorities

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM	Relative Priority
CNN	0.16	0.20	0.19	0.29	0.13	0.19
RNN	0.05	0.07	0.05	0.05	0.09	0.06
LSTM	0.08	0.13	0.10	0.07	0.10	0.10
CNN+RNN	0.08	0.20	0.19	0.15	0.17	0.16
CNN+LSTM	0.63	0.40	0.48	0.44	0.51	0.49

(b) Precision

Table A 3: Accuracy AHP comparison matrix with column sum

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM
CNN	1	4	1/5	1/3	1/4
RNN	1/4	1	1/8	1/6	1/7
LSTM	5	8	1	3	2
CNN+RNN	3	6	1/3	1	1/3
CNN+LSTM	4	7	1/2	3	1
Sum	13.25	26	2.158333333	7.5	3.7261905

Table A 4: Normalized precision comparison matrix with relative priorities

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM	Relative Priority
CNN	0.08	0.15	0.09	0.04	0.07	0.09
RNN	0.02	0.04	0.06	0.02	0.04	0.04
LSTM	0.38	0.31	0.46	0.40	0.54	0.42
CNN+RNN	0.23	0.23	0.15	0.13	0.09	0.17
CNN+LSTM	0.30	0.27	0.23	0.40	0.27	0.29

(c) Recall

Table A 5: Recall AHP comparison matrix with column sum

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM
CNN	1	3	1/2	1/2	1/5
RNN	1/3	1	1/4	1/4	1/7
LSTM	2	4	1	1/2	1/3
CNN+RNN	2	4	2	1	1/3
CNN+LSTM	5	7	3	3	1
Sum	10.33	19.00	6.75	5.25	2.01

Table A 6: Normalized recall comparison matrix with relative priorities

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM	Relative Priority
CNN	0.10	0.16	0.07	0.10	0.10	0.10
RNN	0.03	0.05	0.04	0.05	0.07	0.05
LSTM	0.19	0.21	0.15	0.10	0.17	0.16
CNN+RNN	0.19	0.21	0.30	0.19	0.17	0.21
CNN+LSTM	0.48	0.37	0.44	0.57	0.50	0.47

(d) F1 Score

Table A 7: F1 Score AHP comparison matrix with column sum

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM
CNN	1	3	2	1/2	1/5
RNN	1/3	1	1/2	1/4	1/7
LSTM	1/2	2	1	1/3	1/5
CNN+RNN	2	4	3	1	1/4
CNN+LSTM	5	7	5	4	1
Sum	8.83	17.00	11.50	6.08	1.79

Table A 8: Normalized F1 Score comparison matrix with relative priorities

Alternatives	CNN	RNN	LSTM	CNN+RNN	CNN+LSTM	Relative Priority
CNN	0.11	0.18	0.17	0.08	0.11	0.13
RNN	0.04	0.06	0.04	0.04	0.08	0.05
LSTM	0.06	0.12	0.09	0.05	0.11	0.09
CNN+RNN	0.23	0.24	0.26	0.16	0.14	0.21
CNN+LSTM	0.57	0.41	0.43	0.66	0.56	0.53

Appendix B

Test Results

CNN

Table B 1: CNN test on real world data results

dataset	accuracy	error rate	precision	sensitivity	fallout	specificity	f1 score	MCC
DJIA	0,642	0,357	0,643	0,998	0,999	0,0001	0,782	-0,027
TSLA	0,508	0,491	0,507	0,995	0,994	0,005	0,672	0,011
AAPL	0,466	0,533	0,466	0,992	0,994	0,005	0,634	-0,010
BTC	0,521	0,478	0,521	0,995	0,996	0,003	0,684	-0,006
Stocks	0,657	0,342	0,657	0,999	1	0	0,793	-0,006
Stock Market	0,598	0,401	0,602	0,965	0,950	0,049	0,742	0,044
WallStreetBets	0,591	0,408	0,594	0,987	0,994	0,005	0,742	-0,039
All	0,616	0,383	0,619	0,985	0,980	0,019	0,760	0,024
Average	**0,575**	**0,424**	**0,576**	**0,990**	**0,988**	**0,011**	**0,726**	**-0,001**

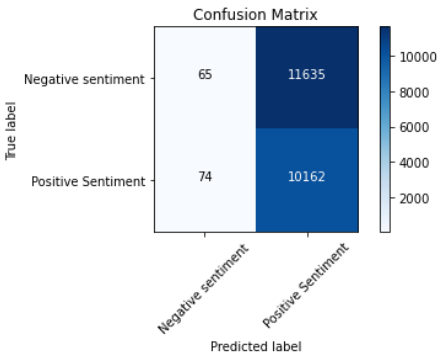


Figure B 1: CNN AAPL confusion matrix

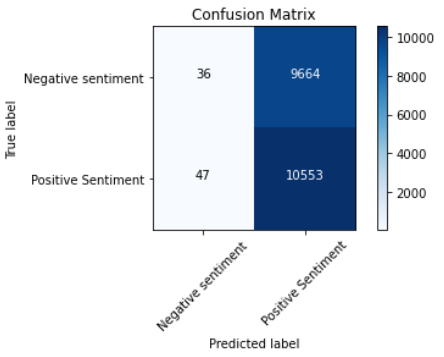


Figure B 2: CNN BTC confusion matrix

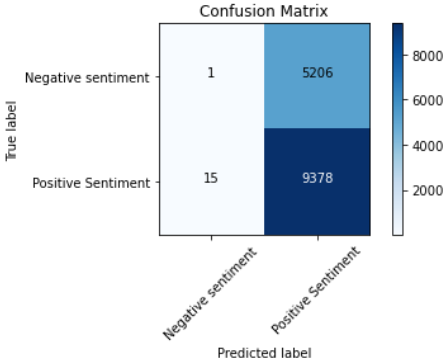


Figure B 3: CNN DJIA confusion matrix

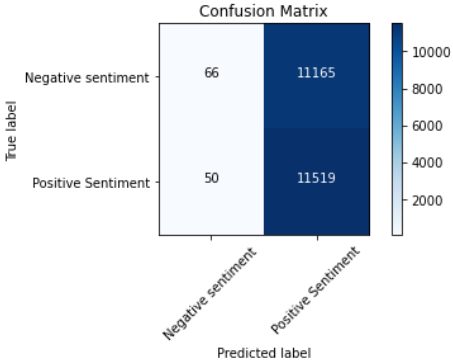


Figure B 4:CNN TSLA confusion matrix

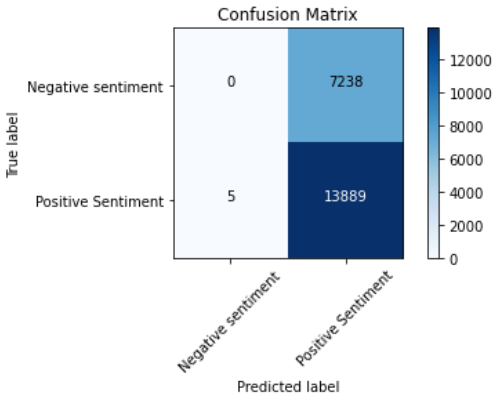


Figure B 5: CNN Stocks w/DJIA confusion matrix

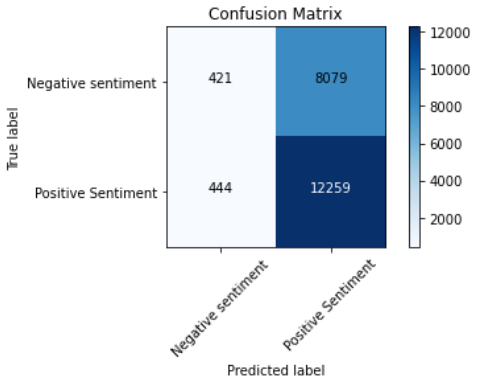


Figure B 6: CNN stock market w/DJIA confusion matrix

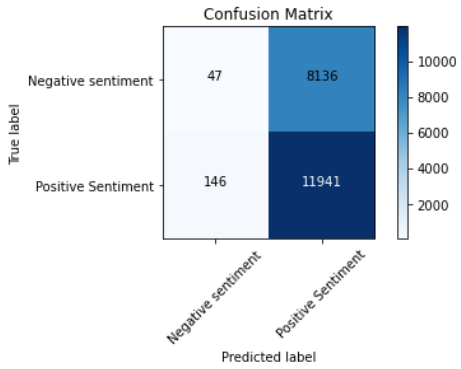


Figure B 7: CNN wallstreetbets w/DJIA confusion matrix

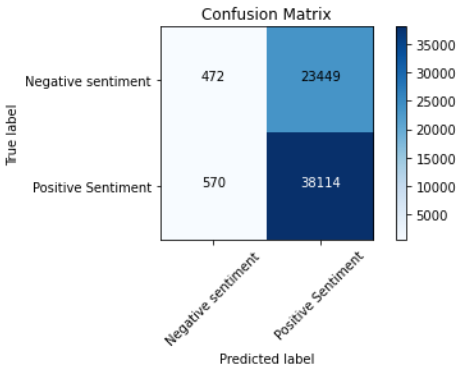


Figure B 8: CNN all reddit data w/DJIA confusion matrix

LSTM

Table B 2: LSTM test on real world data results

datasets	accuracy	error rate	precision	sensitivity	fallout	specificity	f1 score	MCC
DJIA	0,466	0,534	0,607	0,482	0,564	0,436	0,537	-0,086
TSLA	0,512	0,488	0,516	0,692	0,675	0,325	0,591	0,018
AAPL	0,482	0,518	0,465	0,757	0,756	0,244	0,576	0,001
BTC	0,494	0,506	0,514	0,494	0,506	0,494	0,504	-0,012
Stocks	0,514	0,657	0,653	0,554	0,565	0,435	0,600	-0,012
Stock Market	0,551	0,449	0,623	0,635	0,574	0,426	0,629	0,071
WallStreetBets	0,510	0,490	0,589	0,593	0,612	0,388	0,591	-0,021
All	0,525	0,475	0,621	0,593	0,584	0,416	0,607	0,010
Average	**0,507**	**0,515**	**0,573**	**0,600**	**0,604**	**0,396**	**0,579**	-0,004

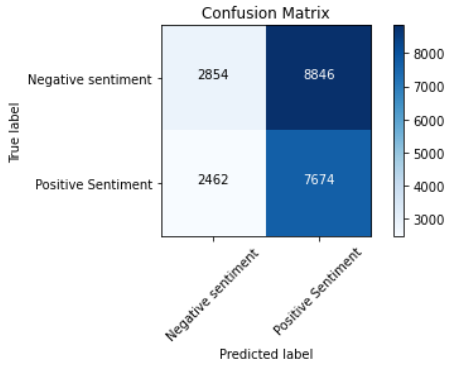


Figure B 9: LSTM AAPL confusion matrix

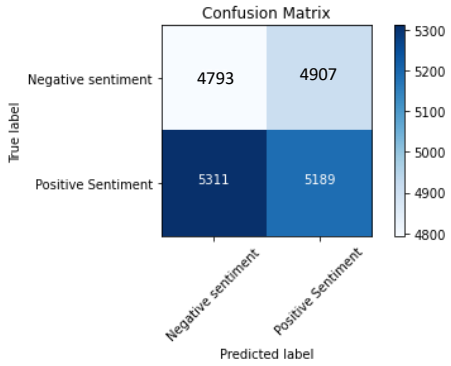


Figure B 10: LSTM BTC confusion matrix

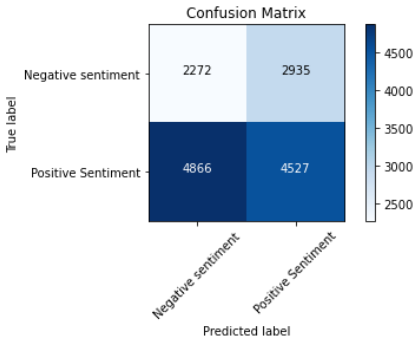


Figure B 11: LSTM DJIA confusion matrix

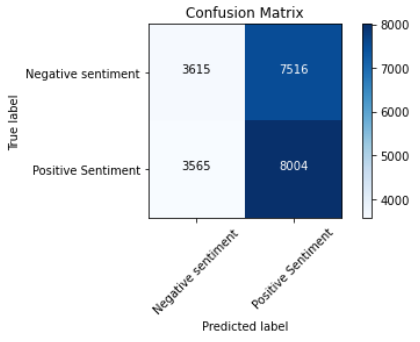


Figure B 12: LSTM TSLA confusion matrix

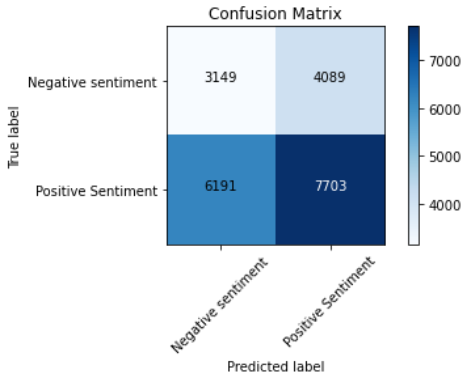


Figure B 13: LSTM Stocks w/DJIA confusion matrix

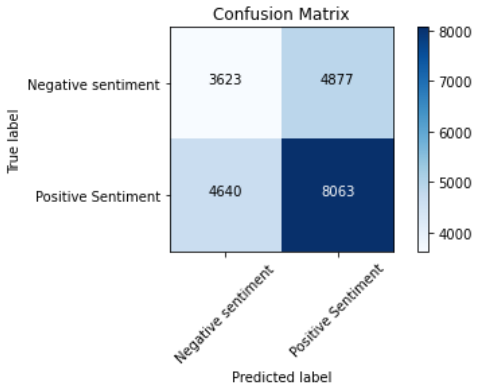


Figure B 14: LSTM Stock market w/DJIA confusion matrix

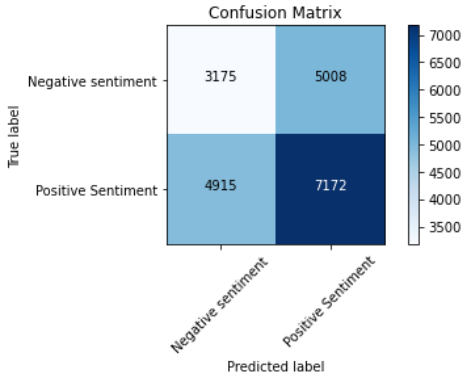


Figure B 15: LSTM wallstreetbets w/DJIA confusion matrix

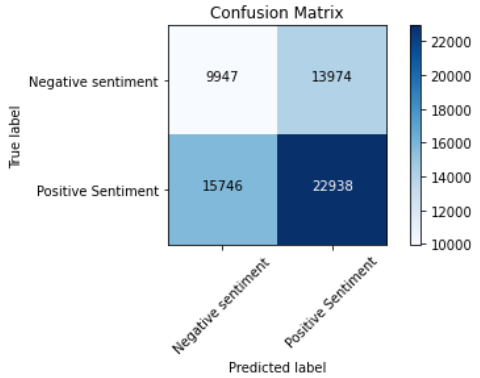


Figure B 16: LSTM all reddit data w/DJIA confusion matrix

GRU

Table B 3: GRU test on real world data results

dataset	accuracy	error rate	precision	sensitivity	fallout	specificity	f1 score	MCC
DJIA	0,368	0,632	0,536	0,137	0,215	0,785	0,218	0,053
TSLA	0,524	0,476	0,522	0,733	0,690	0,310	0,610	-0,008
AAPL	0,493	0,507	0,471	0,691	0,680	0,320	0,560	-0,012
BTC	0,507	0,493	0,523	0,633	0,632	0,368	0,573	-0,017
Stocks	0,533	0,467	0,665	0,583	0,564	0,436	0,621	0,122
Stock Market	0,562	0,438	0,641	0,612	0,513	0,487	0,626	-0,048
WallStreetBets	0,507	0,493	0,590	0,567	0,582	0,418	0,578	-0,034
All	0,552	0,448	0,648	0,601	0,529	0,471	0,624	0,008
Average	**0,506**	**0,494**	**0,574**	**0,570**	**0,551**	**0,449**	**0,551**	**0,008**

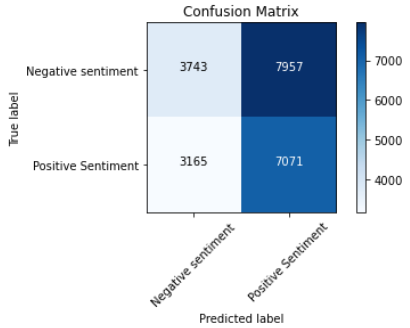


Figure B 17: GRU AAPL confusion matrix

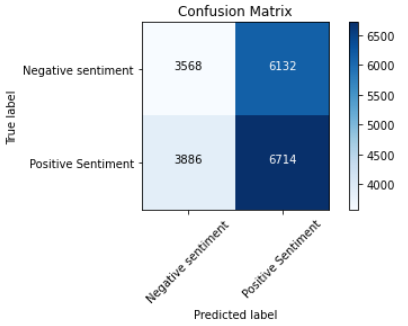


Figure B 18: GRU BTC confusion matrix

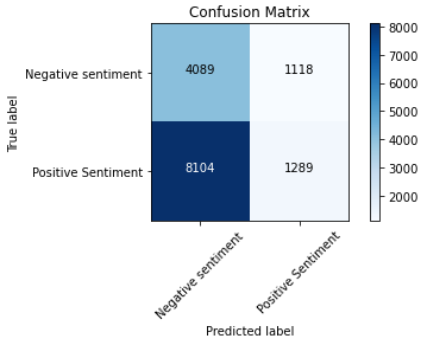


Figure B 19: GRU DJIA confusion matrix

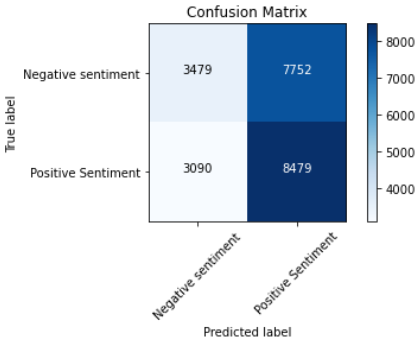


Figure B 20: GRU TSLA confusion matrix

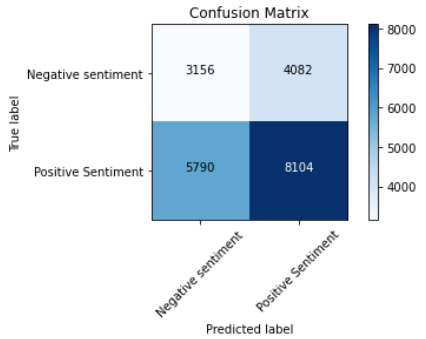


Figure B 21: GRU Stocks w/DJIA confusion matrix

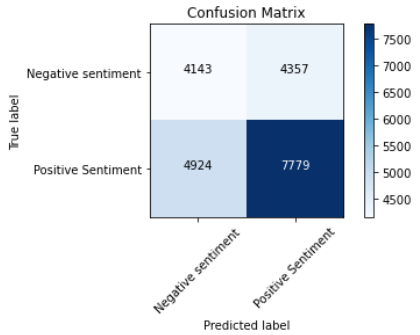


Figure B 22: GRU Stock market w/DJIA confusion matrix

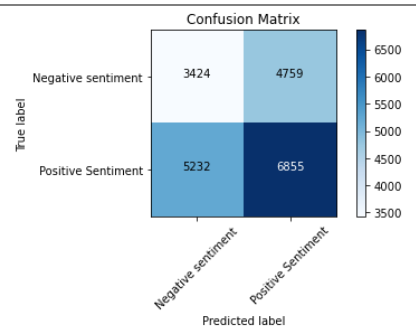


Figure B 23: GRU wallstreetbets w/DJIA confusion matrix

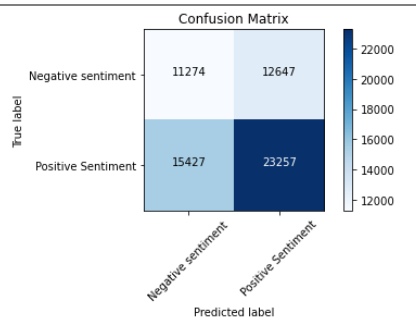


Figure B 24: GRU all reddit data w/DJIA confusion matrix

CNN-LSTM

Table B 4: CNN - LSM test on real world data results

dataset	accuracy	error rate	precision	sensitivity	fallout	specificity	f1 score	MCC
DJIA	0,539	0,461	0,653	0,607	0,583	0,417	0,629	0,028
TSLA	0,515	0,485	0,520	0,565	0,536	0,464	0,542	0,033
AAPL	0,512	0,488	0,482	0,592	0,557	0,443	0,531	0,035
BTC	0,539	0,461	0,653	0,607	0,583	0,417	0,629	-0,028
Stocks	0,540	0,460	0,659	0,620	0,616	0,384	0,639	-0,210
Stock Market	0,517	0,483	0,596	0,602	0,611	0,389	0,599	-0,010
WallStreetBets	0,513	0,487	0,589	0,607	0,625	0,375	0,598	-0,020
All	0,523	0,477	0,615	0,610	0,617	0,383	0,613	-0,008
Average	**0,525**	**0,475**	**0,596**	**0,601**	**0,591**	**0,409**	**0,597**	**-0,023**

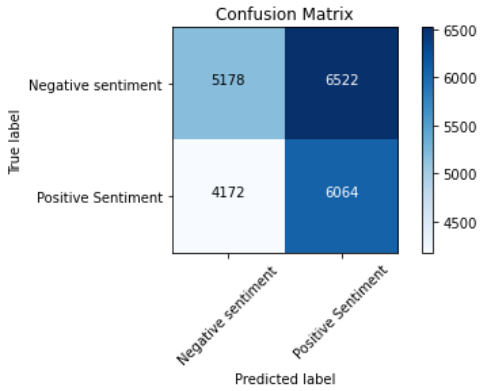


Figure B 25: CNN-LSTM AAPL confusion matrix

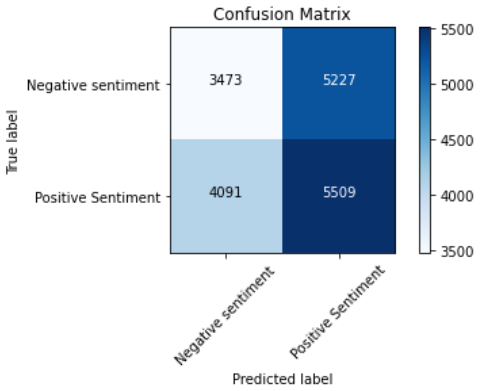


Figure B 26: CNN-LSTM BTC confusion matrix

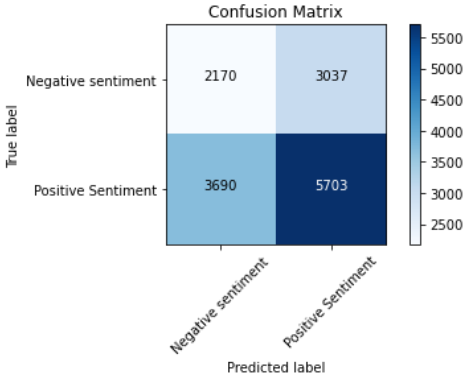


Figure B 27: CNN-LSTM DJIA confusion matrix

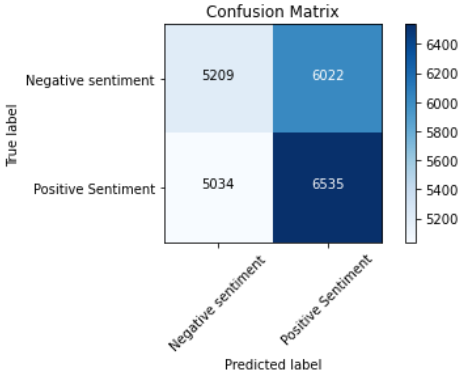


Figure B 28: CNN-LSTM TSLA confusion matrix

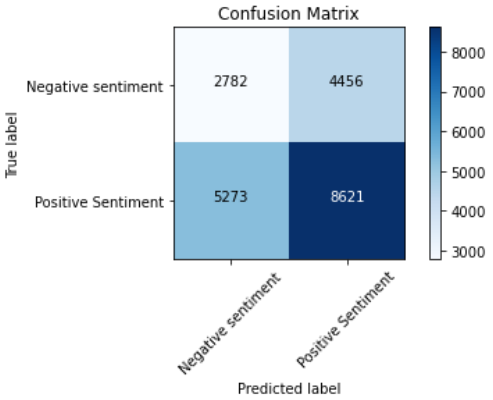


Figure B 29: CNN-LSTM Stocks w/DJIA confusion matrix

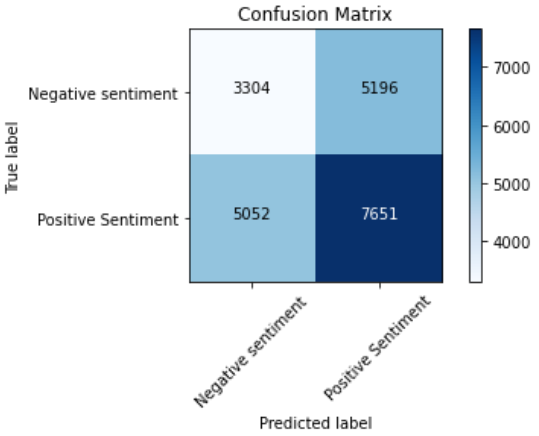


Figure B 30: CNN-LSTM Stock market w/DJIA confusion matrix

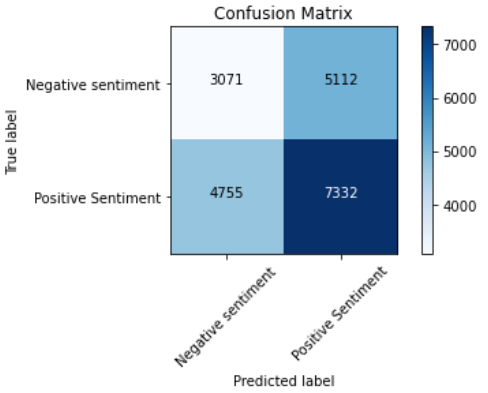


Figure B 31: CNN-LSTM wallstreetbets w/DJIA confusion matrix

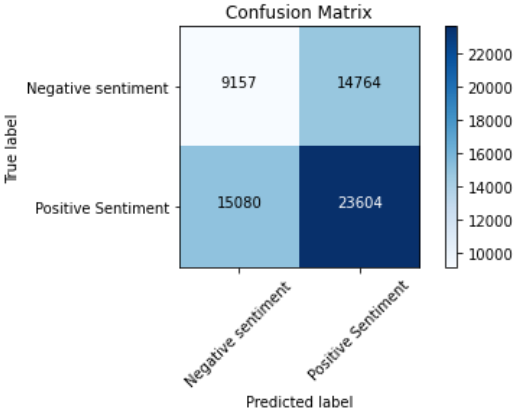


Figure B 32: CNN-LSTM with all reddit data w/DJIA confusion matrix

CNN-LSTM with Stock Indicators

Table B 5: CNN - LSTM with Stock data test on real world data results

dataset	accuracy	error rate	precision	sensitivity	fallout	specificity	f1 score	MCC
DJIA	0,357	0,643	0,333	0,125	0,333	0,667	0,182	-0,237
TSLA	0,643	0,357	1,000	0,375	0,000	1,000	0,545	0,572
AAPL	0,692	0,308	0,750	0,500	0,143	0,857	0,600	0,472
BTC	0,556	0,444	0,625	0,500	0,375	0,625	0,556	0,140
Stocks	0,520	0,480	0,417	0,500	0,467	0,533	0,455	0,030
Stock Market	0,393	0,607	0,143	0,083	0,375	0,625	0,105	-0,280
WallStreetBets	0,333	0,667	0,333	0,111	0,333	0,667	0,167	-0,258
All	0,393	0,607	0,308	0,333	0,563	0,438	0,320	-0,191
Average	**0,486**	**0,514**	**0,489**	**0,316**	**0,324**	**0,676**	**0,366**	**0,031**

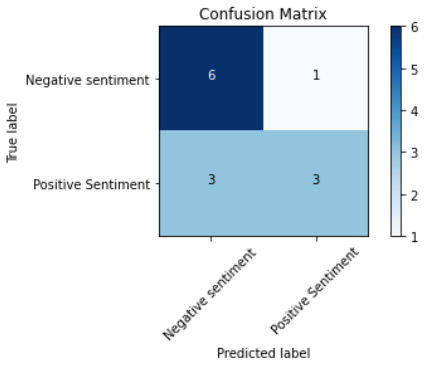


Figure B 33: CNN-LSTM with stock indicators AAPL confusion matrix

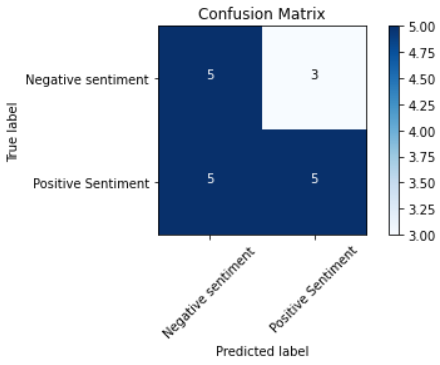


Figure B 34: CNN-LSTM with stock indicators BTC confusion matrix

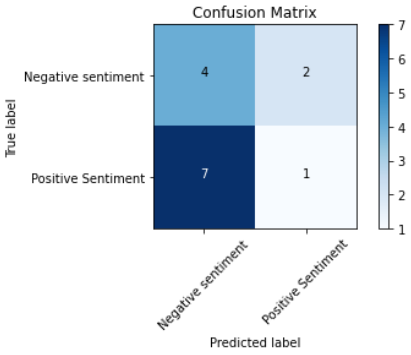


Figure B 35: CNN-LSTM with stock indicators DJIA confusion matrix

Figure B 36: CNN-LSTM with stock indicators TSLA confusion matrix

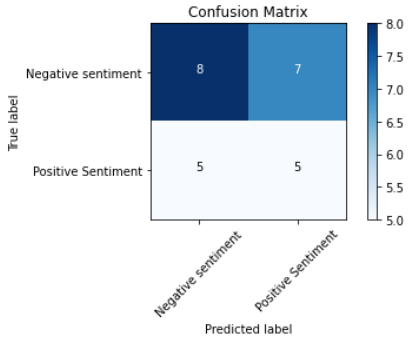


Figure B 37: CNN-LSTM with stock indicators Stocks w/DJIA confusion matrix

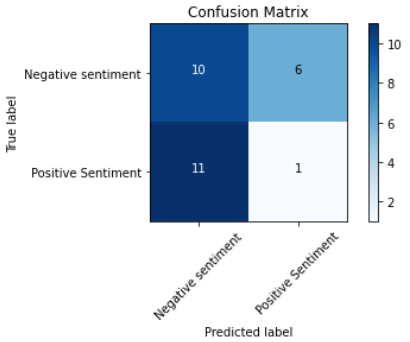


Figure B 38: CNN-LSTM with stock indicators Stock market w/DJIA confusion matrix

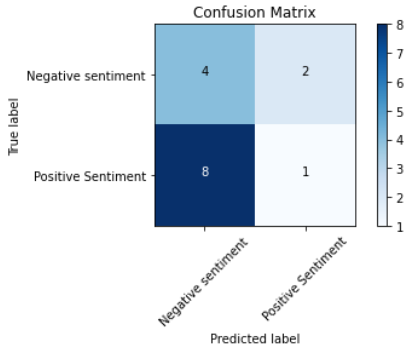


Figure B 39: CNN-LSTM with stock indicators wallstreetbets w/DJIA confusion matrix

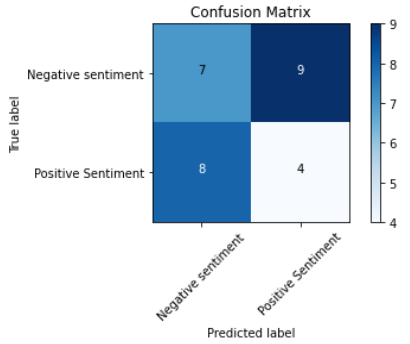


Figure B 40: CNN-LSTM with stock indicators all reddit data w/DJIA confusion matrix

CNN-GRU

Table B 6: CNN – GRU test on real world data results

dataset	accuracy	error rate	precision	sensitivity	fallout	specificity	f1 score	MCC
DJIA	0,563	0,437	0,659	0,665	0,622	0,378	0,662	0,062
TSLA	0,501	0,499	0,505	0,798	0,805	0,195	0,619	0,019
AAPL	0,472	0,528	0,464	0,843	0,853	0,147	0,599	0,020
BTC	0,509	0,491	0,519	0,838	0,851	0,149	0,641	0,020
Stocks	0,581	0,419	0,694	0,648	0,548	0,452	0,670	-0,076
Stock Market	0,507	0,493	0,583	0,624	0,667	0,333	0,603	-0,044
WallStreetBets	0,506	0,494	0,584	0,593	0,624	0,376	0,589	0,001
All	0,532	0,468	0,620	0,623	0,616	0,384	0,622	-0,040
Average	**0,521**	**0,479**	**0,579**	**0,704**	**0,698**	**0,302**	**0,625**	**-0,005**

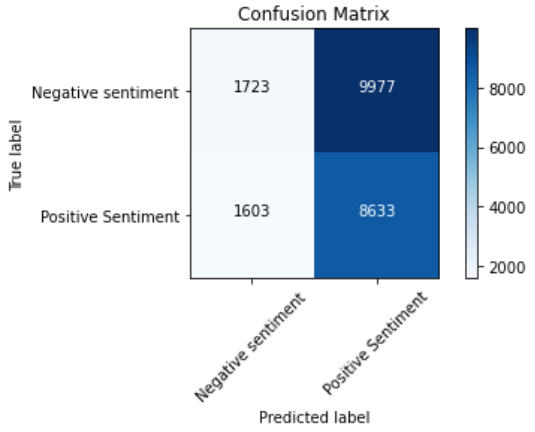


Figure B 41: CNN-GRU AAPL confusion matrix

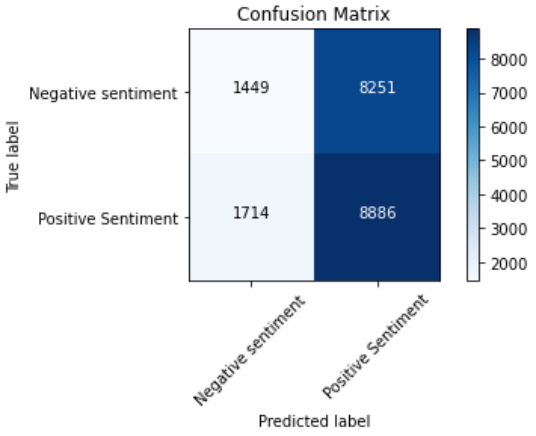


Figure B 42: CNN-GRU BTC confusion matrix

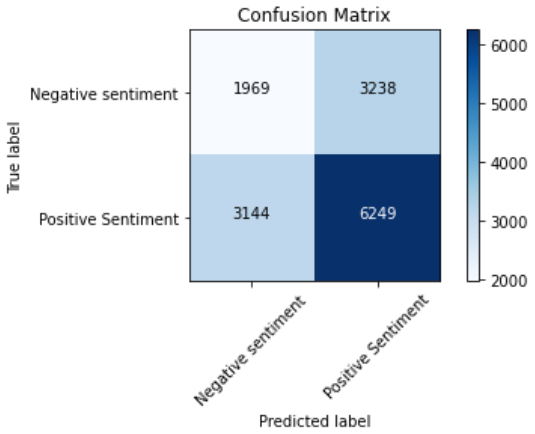


Figure B 43: CNN-GRU DJIA confusion matrix

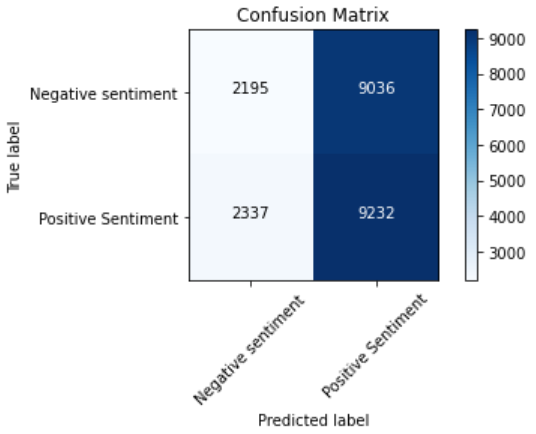


Figure B 44: CNN-GRU TSLA confusion matrix

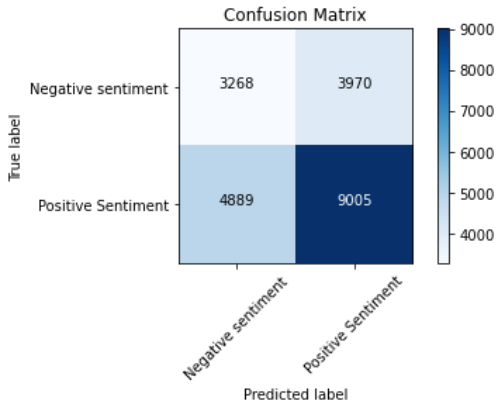


Figure B 45: CNN-GRU Stocks w/DJIA confusion matrix

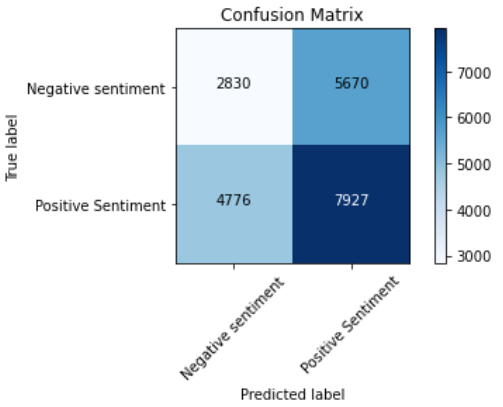


Figure B 46: CNN-GRU Stock market w/DJIA confusion matrix

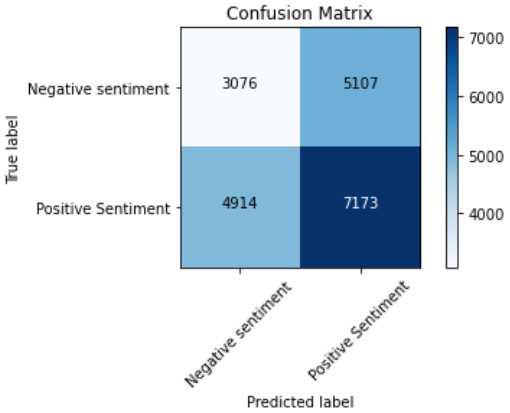


Figure B 47: CNN-GRU wallstreetbets w/DJIA confusion matrix

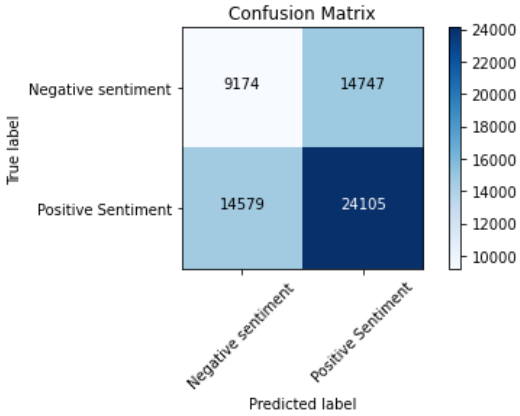


Figure B 48: CNN-GRU with all reddit data w/DJIA confusion matrix

CNN-BiGRU

Table B 7: CNN – BiGRU test on real world data results

dataset	accuracy	error rate	precision	sensitivity	fallout	specificity	f1 score	MCC
DJIA	0,578	0,422	0,659	0,714	0,665	0,335	0,685	-0,101
TSLA	0,511	0,489	0,513	0,719	0,702	0,298	0,599	0,049
AAPL	0,494	0,506	0,473	0,736	0,718	0,282	0,576	0,011
BTC	0,522	0,478	0,530	0,741	0,717	0,283	0,618	0,001
Stocks	0,505	0,495	0,633	0,588	0,655	0,345	0,609	0,022
Stock Market	0,494	0,506	0,582	0,555	0,597	0,403	0,568	0,116
WallStreetBets	0,532	0,468	0,597	0,665	0,664	0,336	0,629	-0,016
All	0,510	0,490	0,604	0,601	0,637	0,363	0,603	0,084
Average	**0,518**	**0,482**	**0,574**	**0,665**	**0,669**	**0,331**	**0,611**	**0,021**

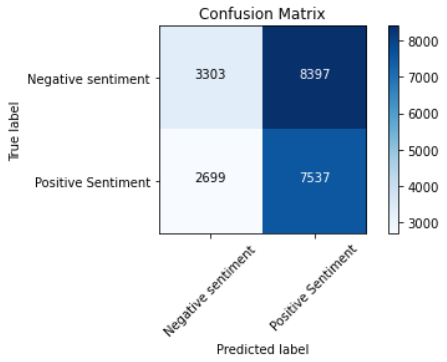


Figure B 49: CNN-BiGRU AAPL confusion matrix

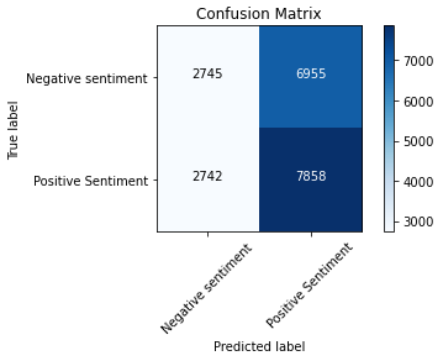


Figure B 50: CNN-BiGRU BTC confusion matrix

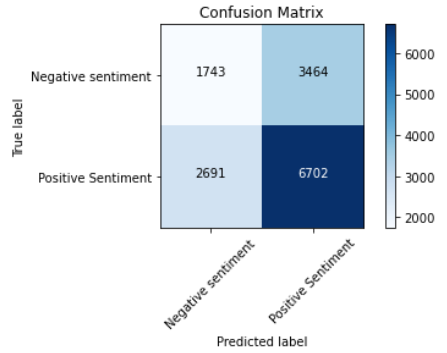


Figure B 51: CNN-BiGRU DJIA confusion matrix

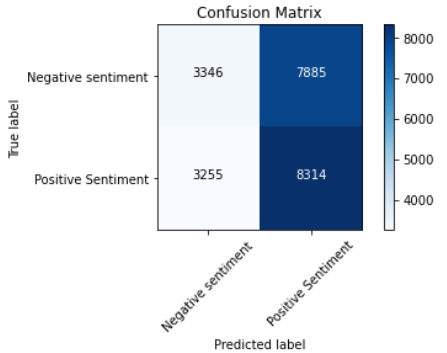


Figure B 52: CNN-BiGRU TSLA confusion matrix

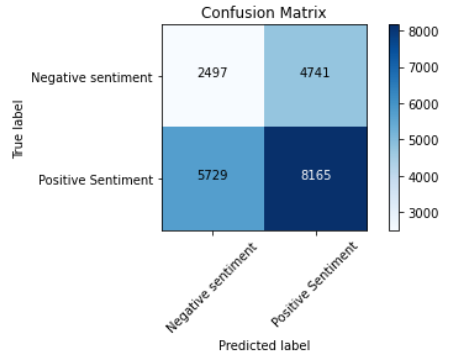


Figure B 53: CNN-BiGRU Stocks w/DJIA confusion matrix

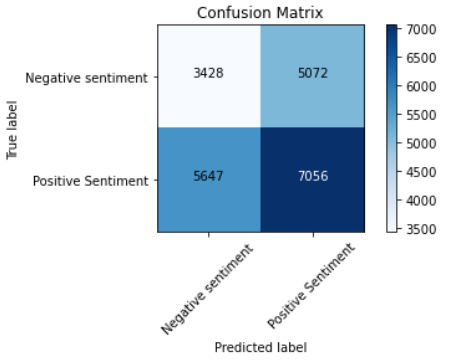


Figure B 54: CNN-BiGRU Stock market w/DJIA confusion matrix

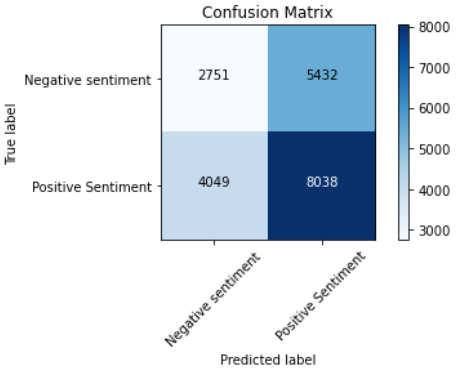


Figure B 55: CNN-BiGRU wallstreetbets w/DJIA confusion matrix

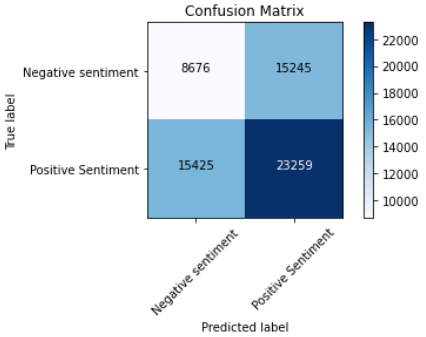


Figure B 56: CNN-BiGRU all reddit data w/DJIA confusion matrix

Simulation Results

TSLA dataset

Table B 8: TSLA simulation result - first week (13/09/2021 to 17/09/2021)

Date	13/09/2021	14/09/2021	15/09/2021	16/09/2021	17/09/2021
Market Open Price	740,214	742,570	745,000	752,830	757,150
Market Closing Price	743,000	744,490	755,830	756,990	759,490
Predicted Sentiment	0	0	0	1	0
Price Change Percentage	0,376	0,259	1,454	0,553	0,309
Invested	0,000	0,000	0,000	1000,000	0,000
Balance	1000,000	1000,000	1000,000	1005,526	1005,526
Savings	1000,000	0,000	0,000	0,000	1005,526
Profit/Loss	0,000	0,000	0,000	5,526	5,526
Baseline	1003,764	1006,359	1020,988	1026,630	1029,803

Table B 9: TSLA simulation result - second week (20/09/2021 to 24/09/2021)

Date	20/09/2021	21/09/2021	22/09/2021	23/09/2021	24/09/2021
Market Open Price	734,558	734,790	743,526	755,000	745,890
Market Closing Price	730,170	739,380	751,940	753,640	774,390
Predicted Sentiment	0	0	0	0	1
Price Change Percentage	-0,597	0,625	1,132	-0,180	3,821
Invested	0,000	0,000	0,000	0,000	1005,526
Balance	1005,526	1005,526	1005,526	1005,526	1043,946
Savings	1005,526	1005,526	1005,526	1005,526	0,000
Profit/Loss	5,526	5,526	5,526	5,526	43,946
Baseline	1023,651	1030,046	1041,702	1039,826	1079,557

Table B 10: TSLA simulation result - third week (27/09/2021 to 28/09/2021)

Date	27/09/2021	28/09/2021
Market Open Price	773,120	787,200
Market Closing Price	791,360	777,560
Predicted Sentiment	0	0
Price Change Percentage	2,359	-1,225
Invested	0,000	0,000
Balance	1043,946	1043,946
Savings	1043,946	1043,946
Profit/Loss	43,946	43,946
Baseline	1105,027	1091,494

AAPL dataset

Table B 11: AAPL simulation result - first week (13/09/2021 to 17/09/2021)

Date	13/09/2021	14/09/2021	15/09/2021	16/09/2021	17/09/2021
Market Open Price	150,630	150,350	148,560	148,440	148,820
Market Closing Price	149,550	148,120	149,030	148,790	146,800
Predicted Sentiment	0	1	1	0	0
Price Change Percentage	-0,717	-1,483	0,316	0,236	-1,357
Invested	0,000	1000,000	985,168	0,000	0,000
Balance	1000,000	985,168	988,285	988,285	988,285
Savings	1000,000	0,000	0,000	988,285	988,285
Profit/Loss	0,000	-14,832	-11,715	-11,715	-11,715
Baseline	992,830	978,104	981,199	983,512	970,163

Table B 12: AAPL simulation result - second week (20/09/2021 to 24/09/2021)

Date	20/09/2021	21/09/2021	22/09/2021	23/09/2021	24/09/2021
Market Open Price	143,800	143,930	144,450	146,650	145,660
Market Closing Price	142,940	143,430	145,850	146,830	146,920
Predicted Sentiment	0	0	0	0	0
Price Change Percentage	-0,598	-0,347	0,969	0,123	0,865
Invested	0,000	0,000	0,000	0,000	0,000
Balance	988,285	988,285	988,285	988,285	988,285
Savings	988,285	988,285	988,285	988,285	988,285
Profit/Loss	-11,715	-11,715	-11,715	-11,715	-11,715
Baseline	964,361	961,011	970,325	971,516	979,919

Table B 13: AAPL simulation result - third week (27/09/2021 to 28/09/2021)

Date	27/09/2021	28/09/2021
Market Open Price	145,470	143,250
Market Closing Price	145,370	141,910
Predicted Sentiment	0	1
Price Change Percentage	-0,069	-0,935
Invested	0,000	988,285
Balance	988,285	979,040
Savings	988,285	0,000
Profit/Loss	-11,715	-20,960
Baseline	979,246	970,086