A COMPARISON OF SENTIMENT ANALYSIS TECHNIQUES IN A PARALLEL AND DISTRIBUTED NOSQL ENVIRONMENT

Dissertation submitted by

IAN DANIËL VAN DER LINDE

Student number: 2010062467

Submitted in fulfilment of the requirements in respect of the Master's Degree

M.Sc. Computer Informatics Systems

in the

Department of Computer Science and Informatics
Faculty of Natural and Agricultural Sciences
University of the Free State, South Africa

09 April 2020

Supervisor: Dr J.E. Kotzé
Co-supervisor: Mr G.J. Dollman

ABSTRACT

Sentiment analysis has seen a revival due to the advent of social media platforms such as Facebook and Twitter. The data posted on these platforms can be mined for valuable insights into customer relations, political unrest and product supply and demand. This information is embedded in typical Big Data, with very large volumes delivered at high velocity consisting of a wide variety of content and sources, and usually unstructured in nature.

The challenge of analysing such data for decision support can be addressed through the use of sentiment analysis techniques in distributed environments designed to process and store large amounts of data in a horizontally-scalable fashion. The performance characteristics of these techniques have, however, hardly been studied in distributed environments, and the impact of cluster size on such environments is largely undocumented.

The aim of this research was to investigate the accuracy and performance of four sentiment analysis approaches (a lexicon-based classifier, a Naïve-Bayes classifier, a Neural Network classifier, and a Support Vector Machine classifier) in a distributed environment with a cluster size of three to eight machines, while making use of a distributed NoSQL database backend to retrieve and store the data.

The key investigations were to determine the nature of performance bottlenecks for each classifier in a distributed environment, how well each classifier scaled as more machines are added, and whether a relationship could be found between classifier accuracy and performance.

It was determined that all four classifiers provide statistically significantly different accuracies, when compared pairwise and collectively. It was also found that there is no clear relationship between accuracy and resource usage (i.e., a more performant technique does not necessarily have worse accuracy).

Keywords: sentiment analysis, NoSQL database, document classification, parallel computing, distributed computing, empirical analysis

OPSOMMING

Sentiment ontleding het onlangs ŉ toename in gewildheid geniet, te danke aan die opkoms van sosiale media soos Facebook en Twitter. Die data wat op hierdie platforms bekendgemaak word kan, onderhewig aan data ontginning, lei tot waardevolle insigte in onderwerpe soos kliënte verhoudings, politieke onrus en die vraag en aanbod van produkte. Hierdie inligting is algemeen versteek in tipiese grootdata: dit kom voor in baie groot volumes, word verskaf teen ŉ hoë snelheid, en is afkomstig van ŉ verskeidenheid van bronne, met verskeie inhoud. Hierdie data is ook tipies ongestruktureerd.

Die uitdagings betrokke by die analise van grootdata soos dié kan aangespreek word deur die toepassing van sentiment analise tegnieke in verspreide omgewings wat ontwerp is om groot hoeveelhede data te verwerk en te stoor op ŉ manier wat horisontaal skaalbaar is. Die werkverrigting van hierdie tegnieke is tot dusver min ondersoek in verspreide omgewings, en die impak van die hoeveelheid rekenaars betrokke is grotendeels nie in literatuur opgeskryf nie.

Die doel van hierdie navorsing was om die akkuraatheid en werkverrigting van vier sentiment ontleding benaderings (‘n leksikon-gebaseerde klassifiseerder, ŉ Naïve-Bayes klassifiseerder, ŉ Neurale Netwerk klassifiseerder en ŉ Ondersteuningsverktor-masjien klassifiseerder) te ondersoek in ‘n verspreide omgewing met van drie tot agt rekenaars, ondersteun deur ŉ verspreide NoSQL databasis vir die stoor en onttrekking van die data.

Die kern ondersoeke van hierdie navorsing was om die aard van die bottelnekke in werkverrigting van elke klassifiseerder te bepaal in ŉ verspreide omgewing, om te ondersoek hoe skaalbaar elke klassifiseerder is soos meer rekenaars bygevoeg word, en of daar ŉ verhouding vasgestel kon word tussen die akkuraatheid en werkverrigting van klassifiseerders oor die algemeen.

Dit was bepaal dat al vier klassifiseerders statisties beduidend verskil in terme van akkuraatheid tydens gepaarde en gekombineerde vergelykings. Dit was ook bevind dat daar geen beduidende verhouding bestaan tussen akkuraatheid en werkverrigting nie (d.w.s. dat ŉ vinniger tegniek nie noodwendig oor swakker akkuraatheid beskik nie).

Sleutelwoorde: sentiment ontleding, NoSQL databasis, dokument klassifikasie, parallelle verwerking, verspreide verwerking, empiriese analise

ACKNOWLEDGEMENTS

The author would like to thank the following persons and entities for their contributions and support:

- My supervisor, **Dr Eduan Kotzé**, for his patience, motivation and valued guidance throughout the process of writing this dissertation.
- My co-supervisor, **Mr Gavin Dollman**, for his thoughtful advice and constructive criticism.
- My former colleagues at the HPC Unit at the University of the Free State: **Mr Stephanus Riekert and Mr Albert van Eck**, for their patient, in-depth training and support, as well as their assistance in terms of equipment.
- My brother, **Jan**, for his extensive assistance with the automation of calculations and generation of graphs reported in this study, in addition to his treasured guidance on other matters concerning programming and the research process.
- My brother's fiancée, **Maheshini**, for her valued advice on the research process, funding applications and proofreading.
- My parents, **Jan and Zelda**, who provided me with the means, motivation and continued support to see this project through to completion.
- Countless other colleagues, friends and acquaintances who kindly provided timely advice and assistance which allowed me to conclude this project.

The author also wishes to give special thanks to the following persons and entities for their substantial financial support:

- **Prof Theo du Plessis** at the **Unit for Language Facilitation and Empowerment** at the University of the Free State.
- The **University of the Free State Tuition Fee Bursary** programme.

The author also wishes to thank the SAS Institute for the use of the SAS University Edition software, which was used to generate the descriptive statistics and graphs in this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

GLOSSARY

API: Application Programming Interface.

Apache Cassandra: A wide-column NoSQL database.

Big Data: Data that is too large to store and analyse using conventional techniques and systems; generally identified based on high velocity, large volumes and a variety of sources or structures.

DNS: Domain Name System.

Empirical analysis: Practical testing and analysis in real-world conditions as opposed to theoretical analysis.

HPC: High Performance Computing.

I/O: Input/output.

Neural Network: A machine learning technique that makes use of artificial neurons structured into layers, with weighted connections between them.

NLP: Natural Language Processing.

NoSQL database: Not only SQL: A database that does not make use of a relational database structure.

NTP: Network Time Protocol.

OAI-PMH: Open Archives Initiative Protocol for Metadata Harvesting.

Support Vector Machine: A machine learning approach that classifies data according their position in n-dimensional vector space in relation to an $(n-1)$-dimensional hyperplane.

SVM: See: Support Vector Machine.

TSA: Twitter Sentiment Analysis. See also: Twitter

Twitter: A microblogging social media service.

UGC: User-generated text.

CHAPTER 1
INTRODUCTION

1.1 Introduction

The growth of data increases by roughly ten times every five years and is expected to result in an estimated forty zettabytes by 2020. This represents an estimated 5.2 terabytes of data per person, globally (Del Vecchio, Di Minin, Petruzzelli, Panniello, & Pirri, 2018). It is expected that this figure will continue to rise as time passes, resulting in continued challenges pertaining to the storage and analysis of such data, which presents great value and opportunity.

These challenges arise primarily due to the unstructured nature, variability and high volumes associated with Big Data (Alam, Muley, Kadaru, & Joshi, 2013). For businesses to retain and improve their position in competitive markets, they need the capacity to store, process and retrieve this data for analytics purposes (Davenport, 2006). The most prevalent systems used for data storage and analytics today are relational database systems from vendors such as Oracle, Microsoft, IBM and SAP (Chen, Chiang, & Storey, 2012; Moniruzzaman & Hossain, 2013). The current issue is that these relational database systems do not have the capacity to take full advantage of the volumes of streaming data to improve aspects such as their customer relations, market intelligence and recommendation systems (Chen et al., 2012; Del Vecchio et al., 2018; Moniruzzaman & Hossain, 2013).

The primary effort to resolve the problem of Big Data storage comes in the form of NoSQL, or “Not only SQL” databases, which are known for their ability to scale and manage distributed datasets and are accepted as the current standard for doing so, especially in the case of unstructured social media data (Moniruzzaman & Hossain, 2013). Several different types of NoSQL databases have been developed for various purposes (e.g. high-speed distributed storage and graph analysis) and environments, but they generally have overlapping features such as horizontal scalability and fault tolerance. It is generally accepted that storage and analysis for Big Data projects cannot be supported by traditional SQL databases, and instead require highly scalable and distributed NoSQL databases (Krishnan, 2013). Social media, as a form of Big Data (Boyd & Crawford, 2012), was of particular interest to this study, and as such made use of NoSQL technology to serve as a storage backend.

Social media can provide unparalleled access to vast amounts of public consumer opinions, which has the potential to allow businesses to refine their products and services according to the needs of their clients without resorting to limited surveys and guesswork. Additionally, companies or organisations can identify shortcomings with their service and product offerings, and also that of their competitors due to the public nature of this data. Having the capability to process this valuable social media data in real time thus presents businesses with a potential competitive advantage.

Sentiment analysis, also called *opinion mining*, is a very popular natural language processing (NLP) application used to mine and extract meaning from user generated text (UGC) such as social media posts, Tweets, product reviews and blogs. The main goal of sentiment analysis is to determine whether a text, or part of it is subjective or not, and if subjective, whether it expresses a positive or negative viewpoint (Taboada, 2016). There are a number of existing techniques (e.g. Naïve Bayes, latent semantic indexing, decision trees, term frequency-inverse document frequency, expectation maximization, artificial Neural Networks and Support Vector Machines) that can be used to extract sentiment, or opinions, algorithmically from text data, depending on a number of factors such as the domain and length of the data. Sentiment analysis techniques generally have higher accuracy when trained for a particular domain and length (Pang & Lee, 2008). Sentiment analysis can be performed on sentence, paragraph or document level to learn the polarity of words or phrases (Pang & Lee, 2008). There are some examples in literature of accuracy comparisons between various analysis techniques when applied in single-machine computing environments (Agarwal, Xie, Vovsha, Rambow, & Passonneau, 2011; Kouloumpis, Wilson, & Moore, 2011), but with the increasing trend of larger and larger data sets, it becomes unfeasible to rely only on the processing power of single-machine computers. Minelli, Chambers, & Dhiraj (2013) noted that Big Data processing requires parallel computing distributed across multiple computers, but the impact of a distributed architecture on the performance of sentiment analysis algorithms has not been documented adequately.

It could be argued that making use of distributed systems can significantly increase the costs associated with data processing, but using software that provides the ability to scale outwards (i.e. improving processing capacity by adding more machines) could prove to be less expensive

than scaling upwards (improving performance by upgrading a single machine). The reasoning for this is that multiple low-cost commodity machines can provide similar performance to a single enterprise machine, at a reduced price point (at the loss of enterprise characteristics such as improved power consumption and lower failure rates). Additionally, scaling only upwards is no longer desirable since it provides no redundancy and thereby creates a single point of failure. This study made use of multiple low-cost, second-hand desktop machines for all experiments, to ensure that the results are applicable to a wider set of use cases. Limited research (Khuc, Shivade, Ramnath, & Ramanathan, 2012; Stewart & Singer, 2012) has been done to investigate the performance differences between single-machine and cluster computing for analysis, but does not include any details apart from accuracy and running time. In addition to this, there is a lack of documented resource requirements for these algorithms, even though figures such as memory usage, power draw and disk I/O (input/output) should be known during the planning and procurement phases of such projects. This study focused on lexicon-based, Naïve-Bayes, artificial Neural Networks and Support Vector Machines.

The natural progression of this was to make use of a combination of distributed computing, NoSQL and sentiment analysis techniques to provide a robust platform for reliable social media analysis that is easily scalable, fault-tolerant and accurate according to the needs and resources of the users.

1.2 Aim of research

The aim of this research was to provide a comprehensive comparative benchmark between four sentiment analysis approaches in a cluster environment with eight machines. All tests started with three machines with increments of one machine until all eight machines are utilised. The sentiment analysis techniques that were compared to each other were: lexicon-based, Naïve Bayes, a feed-forward artificial Neural Network and a linear Support Vector Machine. It was expected that such a comparison will prove to be a useful guideline during the planning phases of Big Data projects that involve sentiment analysis of social data. The expected results could serve as an indication as to what kind of resources each algorithm requires.

All of the experiments conducted in this study used real-world data which was streamed from Twitter. The experiment was conducted in a controlled laboratory environment making use of

commercial-off-the-shelf server hardware and an open source NoSQL database backend. This controlled environment allowed the expected results to be of maximum benefit to the widest possible audience, given that all of these resources are generally available to businesses of any size and does not require any custom or specialised equipment. Additionally, the benchmarks ran on an eight-machine cluster, which was expected to be sufficient to illustrate the diminishing returns from added machines (if any). From that it was expected to be possible to extrapolate, within reason, the performance of the algorithms and databases beyond the tests that are currently planned.

1.3 Problem statement

Research surrounding the performance (especially in terms of aspects such as speed and resource usage) of sentiment analysis algorithms is limited, and generally do not consider metrics beyond accuracy for comparison. This presents a gap in current literature that requires every researcher or business to first experiment with various sentiment analysis techniques to determine which one provides the accuracy and performance that they require. This study documented some of the factors involved in such a decision in an attempt to improve this gap in literature by doing the following:

Test four sentiment analysis algorithms empirically. The four algorithms were be: Lexicon-based, Naïve-Bayes, a feed-forward artificial Neural Network and a Support Vector Machine. Each algorithm was be tested in turn, and the performance metrics of all the machines was recorded throughout.

For empirical testing, benchmark tests were coducted and four sentiment analysis algorithms were compared in terms of i) accuracy, ii) throughput, iii) power consumption, iv) memory usage, v) network impact and vi) training time. The experiment was conducted in a parallel, distributed environment consisting of eight machines making use a NoSQL database backend.

1.4 Research questions

This study aimed to investigate and answer the following research questions through the use of an empirical experiment:

RQ_1: Can bottlenecks in classifier performance be addressed by increasing the number of machines?
RQ_2: How does the number of machines used affect the data throughput per machine?
RQ_3: Is there an inverse correlation between the throughput and accuracy of the algorithms for each metric (i.e. does a faster algorithm have poorer accuracy)?
RQ_4: Is there an inverse correlation between the training time and training resource usage, and accuracy (i.e. does a less accurate algorithm train faster)?

1.5 Research objectives

This section lists the research objectives (both theoretical and empirical) for the study. These objectives will be discussed and reviewed again throughout this text and summarised again in the concluding chapter.

RO_1: Determine the gap in current literature (theoretical)
RO_2: Compare the resource usage of four sentiment analysis approaches (empirical)
RO_3: Compare the classifier performance of four sentiment analysis approaches (empirical)

1.6 Importance of the research

It can be expected that the amount of data that requires storage and processing will only increase as time goes on. This presents a continuous problem of capacity to aggregate such data in a way that could meaningfully support decision making, such as the case presented earlier regarding the application of sentiment analysis on social media Big Data to perform tasks such as customer support and market analysis.

A certain level of planning, knowledge and expertise is required to apply such techniques effectively in the type of environment that possesses the horizontal scalability necessary for the analysis of Big Data, and among that is the knowledge regarding the behaviour of each technique in a distributed environment. This includes aspects such as the amount of memory that a classifier implementation of a certain technique might require given a certain input, or

the amount of time it takes to train a classifier given an input dataset with particular characteristics.

Literature on practical specifics such as this is scarce, and that leads to perpetual repetition of the same experimentation to determine the real-world characteristics of such approaches that is required for thorough planning of infrastructure and costs pertaining to a particular analysis need. This lack of general availability of such information presents challenges not only to commercial ventures, but also researchers who aim to replicate and improve upon existing studies. It is common for studies to report that certain approaches, for example, run out of memory (or take too long to complete) under specific circumstances, without mentioning any concrete figures that would give the reader an indication of what they could expect should they want to repeat the experiment.

The lack of such specifics has a particular impact on researchers who do not have access to on-demand, scalable computing (such as the public cloud), since there is no indication of what the replication or expansion of a certain study's results might require. This makes it difficult to plan and finance research in this area without some degree of risk that whatever is planned might not be sufficient to test the hypothesis.

This study reports on metrics obtained from commodity hardware during the training, evaluation and testing of a number of sentiment analysis approaches at various levels of distributed computing, with measurements taken at cluster sizes of three to eight machines (with a ninth machine providing support services without directly participating in experimental runs). The results of two hundred experimental test runs was documented, along with forty evaluation runs and forty training runs, in an attempt to fill the void of empirical data (especially in terms of resource requirements) which currently exists in literature.

1.7 Research design

This study followed a positivist experimental approach. A sentiment classification cluster will be created to generate quantitative performance data in order to gain empirical results to compare four sentiment analysis algorithms in a distributed environment. This study will make use of real-world streaming Twitter data for the purposes of the performance benchmark.

1.8 Research methodology

A number of different research methodologies exist, but experiments are generally applied in cases where measurable observations can be made to compile quantitative data for analysis. Experiments allow for the investigation of cause and effect through the use of hypotheses and research questions. This generally involves the identification of dependent and independent variables prior to the experiment: the independent variables are then modified in order to study the effect that this has on the dependent variables in an attempt to establish a causal relationship (Oates, 2005).

Experiments can generally be categorised as either formal experiments or field experiments. Field experiments take place in an uncontrolled environment that generally reflects real-world conditions. Such conditions generally prevent the researcher from accounting for the influence of all the independent variables since they may not be under his or her control. This reduces the scientific rigour. The advantage of a field experiment is that, since the conditions are a good reflection of what happens in the real world, the results of the experiment may more accurately reflect the results of a real-world scenario (Mouton, 2001; Oates, 2005).

This study made use of a formal experiment as research instrument. Such an experiment attempts to more closely control all the variables that may impact the dependent variables, and generally makes use of more artificial laboratory conditions (Mouton, 2001). This is possible for this study given that the equipment used can be isolated from external factors while still retaining the general hardware and software environment in which analysis of this nature is often done.

The empirical analysis was performed as follows: a distributed system was developed to perform sentiment analysis on Twitter data. The system consisted of two parts: a data gathering component which connected to a Twitter stream to download the data necessary for the experiment, and a sentiment analysis component which was responsible for performing sentiment analysis on the data. Twitter data downloaded from the Twitter Streaming API represents a random sample of up to 1% of Tweets worldwide.

The data gathering component was written in Go (Donovan & Kernighan, 2015), for maximum performance in order to keep up with the data stream on limited computing resources, without falling behind. The sentiment analysis component was written in Python, since there are already many natural language processing libraries available for Python, and it can generally be expected to be applied for real world use in most cases because of this. It was also necessary to implement all the algorithms in the same programming language to provide a meaningful comparison, which makes Python the ideal candidate.

Twitter was used as a streaming data source to benchmark each algorithm's performance independently (i.e. no two algorithms will be tested at the same time). Each test started with three machines, and one machine was added after each test until all eight machines took part. Training data was constructed automatically using emoticons. All benchmarks was performed on Apache Cassandra (Datastax, 2018a), a wide-column NoSQL database that can be distributed over an arbitrary number of machines.

The quantitative data was collected directly from each machine involved in the experiment and sent to a central server for storage and analysis. The metrics that were collected were as follows: accuracy (based on precision and recall), throughput, power consumption, memory usage, network utilisation and training time. This data was then aggregated and summarised using SAS to compare the different sentiment analysis approaches. SAS was also used to generate the graphs in this study.

1.9 Research environment

The research environment consisted of nine Dell Optiplex 990 desktop machines connected to a 100 Mbps network switch to facilitate communication. Each machine had a quad core Intel Core i5 processor running at a base clock speed of 2.4 Ghz, with four gigabytes of RAM. A one gigabit per second shared internet connection was used to connect to the Twitter streaming API.

A nine machine cluster was chosen to allow for a meaningful performance comparison as more machines are added during experimentation, similar to the work done by Khuc et al. (2012) which already indicated performance trends using only five machines. This was expected to

clearly illustrate the concept of diminishing returns per machine as the cluster grows larger, and give an indication of what to expect when expansion exceeds nine machines.

1.10 Scope and limitations of the study

On a methodological level this study aimed to investigate the four chosen sentiment analysis approaches in a purely practical manner (i.e. through empirical comparison) without attempting to analyse the theoretical algorithmic complexity or the underlying theoretical causes of the empirical results. This study was therefore limited to practical analyses and comparisons between the approaches and their software implementations.

From a practical standpoint, this study was limited in terms of: the number of machines available to partake in the experiment (therefore limiting the scalability of the experiment), the time constraints under which the study took place, which limits the number of experiments (and the length of an individual experiment) that can take place, and equipment limitations that prevented the measurement of certain metrics commonly associated with empirical studies.

1.11 Contribution

- This study measured the accuracy and performance of four popular sentiment analysis algorithms in a distributed environment. The possibility of a relationship between accuracy and performance was investigated.
- The study measured the resource impact of each algorithm on infrastructure (i.e. how *resource hungry* is each algorithm?) To the author's knowledge, this has not been done previously in literature.
- The study measured the impact of a distributed architecture on performance, and the possible diminishing return on investment as machines are added. This was done by Khuc et al. (2012), but did not include the same variety of metrics.
- It should also be possible to replicate the structure of this study to expand the research to include other sentiment analysis algorithms, and even to serve as a framework for comparing other types of algorithms to investigate how suitable they are for distributed parallelisation.

1.12 Structure of the dissertation

This dissertation is structured as follows:

Chapter two explores the existing literature surrounding Big Data analytics before discussing sentiment analysis and NoSQL storage in more detail. This chapter also discusses the gaps in current literature which leads to the purpose of this study.

Chapter three discusses the possible research paradigms and explains the motivation for choosing a particular paradigm and research design. This chapter also includes a discussion of the formal experiment as a methodology and uses this a basis for the research process that follows. The analysis and interpretation of the data obtained is also briefly discussed prior to examining the limitations of this study.

Chapter four covers the research environment in terms of hardware and software. The responsibilities of the machines at a high level is presented before moving on to the software aspect. The software is discussed in two parts: commodity software supporting the experiment, and research software responsible for running the experiment itself.

Chapter five discusses the experimental design and methodology as a deviation from traditional sentiment analysis processes due to a difference in outcomes and presents the alternative data pipeline in detail from data collection to reporting. This chapter also covers empirical analysis in detail to establish the metrics commonly used in experiments of this nature, and explains their application in this study.

Chapter six presents the results of the study from a number of perspectives. For each sentiment analysis approach, an overview of metrics collected during training, validation and testing is given at every step of the experiment. For testing, the metrics are discussed at every cluster size as well. The results from all four approaches are then presented together comparatively in terms of training and testing metrics.

Chapter seven serves as a summary of the study and a discussion of the results obtained in chapter six. This chapter also includes an overview of the limitations of this study, the possibilities for future research and the final conclusion.

1.13 Summary

This chapter provided an overview of the background of the research problem and the aims of this study to fill the gaps in current literature. This chapter also provided an overview of all the remaining chapters in the dissertation. The following chapter will discuss the challenges associated with Big Data analytics and the introduction of NoSQL as a means to store and process it, as well as sentiment analysis as a means of Big Data analysis.

CHAPTER 2
BIG DATA ANALYTICS

2.1 Introduction

Chapter 1 provided an overview of the research problem and the aim of this study as a means to determine an optimal choice of sentiment analysis algorithm based on a number of factors. This chapter discusses the existing literature surrounding the challenges associated with Big Data and the advent of NoSQL databases in conjunction with sentiment analysis (as a form of Big Data analysis) as a possible solution to some of these challenges. This chapter, therefore, represents the literature review part of the research process within the dissertation, and informs the methodology and experimental design steps within the context of the research questions.

The terminology and characteristics of Big Data, as an emerging phenomenon resulting from the surge in data generated globally, are discussed in detail, including the original three V's (Volume, Velocity and Variety), and an additional three V's (Veracity, Variability and Value) as proposed by subsequent publications. These characteristics form the basis of the need for NoSQL (i.e. not only SQL) databases. This type of distributed database is subsequently discussed in the context of its advantages and disadvantages, in conjunction with the various types available. The characteristics of these databases are discussed individually, followed by a summary of the CAP theorem for distributed databases, as a means to place NoSQL technology in context.

Big Data analytics will then be introduced as a way of addressing the challenges surrounding the processing aspect of Big Data, before discussing the various kinds of text mining that can be applied in this context. A detailed study of sentiment analysis follows, with a discussion of the various approaches and their implementations. Following this, an overview of the applications of sentiment analysis is given prior to a more in-depth discussion of the applications surrounding Twitter sentiment analysis, as part of one of the three high-level approaches. An analysis of the current gap in the literature discussed is then performed within the context of the aim of this study to inform the research process going forward.

2.2 Big Data

Big Data is a term used to describe the recent surge in the speed and volume at which data is generated and collected (Alam et al., 2013; Tsai et al., 2015). Social media data can be the result of a number of sources, such as social media data created by users of services (i.e. user generated content) such as Facebook (Facebook, 2020), Twitter (Twitter, 2020), Instagram (Instagram, 2020) and Snapchat (Snapchat, 2020), or more technical data from clickstream analytics, webserver logs, and firewall data. The advent of the Internet of Things (IoT), which involves connecting traditionally offline appliances (e.g., televisions, fridges, light switches and cars) and machinery (e.g., power stations, dams and factories) to the internet, will also contribute a considerable amount of data which needs to be analysed and stored. Such vast amounts of data cannot be processed using conventional systems that scale vertically, and require distributed computing and storage solutions to accommodate.

2.2.1 The six V's

The term Big Data is not clearly defined, but there are a number of characteristics that are generally used to describe it, usually in terms of V's. The most common of these are the three V's: Volume, Velocity, and Variety (Alam et al., 2013). These more common V's were then later supplemented by Veracity, Variability, and Value.

Volume refers to the amount of data involved, and is usually given in terms of petabytes or more. It represents a challenge in terms of data storage, and may require multiple levels of storage in operational systems and archival systems such as data marts and warehouses. Ensuring the integrity and reliability of such data is also difficult as it may require replication and regular, automatic checksums across many machines and different kinds of storage media.

Velocity is the speed at which the data is generated. It can involve thousands of records or events per second which needs to be processed and stored in real time, and requires redundant, horizontally scalable hardware and software to reduce downtime and keep up with the constant torrent of new data (Alam et al., 2013).

Variety represents the convergence of multiple, disparate data sources and the challenges involved in combining this data in such a way that valuable insights can be extracted from it for decision making purposes (Alam et al., 2013).

As Big Data became more commonly known, a number of authors suggested the addition of several other V's (Bagga & Sharma, 2018), such as Veracity, Variability (or Volatility) and Value.

Veracity refers to the quality of the data received or generated, and can influence the accuracy and reliability of the analysis and insights gained from it. This is key, since poor input data usually results in poor output, something that the processing or analysis component cannot make up for.

Variability (also called Volatility) represents changes in meaning or importance of data. This is especially important in systems that deal with natural language, where the meaning of a word in social media communities can change quickly. In other cases, the meaning of a term might not change, but the associations with it might change, which should be taken into account during analysis.

Value refers to the worth that can be extracted from the data (Bagga & Sharma, 2018; Khan, Uddin, & Gupta, 2014). This can be influenced by the other factors, such as the volume and veracity.

Several solutions have been presented as a means to accommodate and manage Big Data, usually in the form of a NoSQL software stack for analysis and storage. Section 2.3 discusses various NoSQL databases, their features and shortcomings, and why NoSQL is used instead of relational databases for this task.

2.3 NoSQL

NoSQL, or "Not only SQL" databases are known for their ability to scale and manage distributed datasets and are accepted as the current standard for storing and processing Big Data (Cattell, 2011). Several different types of NoSQL databases have been developed for

various purposes and environments namely: key-value stores, document databases, wide-column databases and graph databases (Moniruzzaman & Hossain, 2013). The popular databases among these categories are MongoDB, Redis, Apache Cassandra, Apache HBase, Neo4j, Apache Accumulo and Riak (Moniruzzaman & Hossain, 2013). Wide-column databases like Cassandra and Accumulo are usually write-optimised to ensure optimal performance for data storage (Dede, Sendir, Kuzlu, Hartog, & Govindaraju, 2013). Accumulo and Cassandra are of particular interest to this study due to their high performance, which makes them especially suitable for storing streaming data. These databases also require distributed computing infrastructure such as clusters or grids to perform well in a Big Data environment (Minelli et al., 2013).

2.3.1 Categories

NoSQL databases, unlike relational databases, can have a very diverse set of data models. These data models can be divided into four broad categories of NoSQL databases: Document databases, wide-column (or columnar) databases, key-value stores and graph databases (Gessert, Wingerath, Friedrich, & Ritter, 2017; Gupta, Tyagi, Panwar, Sachdeva, & Saxena, 2017; Moniruzzaman & Hossain, 2013).

Document databases such as MongoDB and CouchDB are aimed at storing documents in various formats such as JSON and XML. This allows them to store semi-structured data in the form of multiple attribute/value pairs. Their ability to perform searches on both the attributes and the values sets them apart from key-value stores which can only search on keys (Gessert et al., 2017; Gupta et al., 2017; Moniruzzaman & Hossain, 2013).

Wide-column databases (also called column stores or columnar databases) make use of a columnar data representation that allows multiple attributes to be stored for every key. These data structures are usually partitioned to permit for the data to be distributed across multiple instances, and are generally well-suited for large-scale storage and processing. Some examples of wide-column databases are Google BigTable, Apache Cassandra and Apache Accumulo (Gessert et al., 2017; Gupta et al., 2017; Moniruzzaman & Hossain, 2013).

Key-value stores operate using a simple data model involving unique keys and associated values (which can usually be any basic data type or even lists of values). This data model only

allows searches to be done on exact keys, and not values. This simplicity makes key-value stores very fast and easily scalable. Redis, DynamoDB and Voldemort are examples of key-value stores (Gessert et al., 2017; Gupta et al., 2017; Moniruzzaman & Hossain, 2013).

Graph databases rely on a mathematical graph model that involves nodes and edges (Gupta et al., 2017). These nodes represent objects or records, and the edges between them represent inter-object relationships. This representation makes them useful for situations where large sets of interconnected objects are concerned, such as with social media (networks of followers or friends) or documents with references to other documents, as is the case with academic papers (Moniruzzaman & Hossain, 2013). Neo4j and Giraph are two examples of databases that make use of a graph model (Gessert et al., 2017; Guo, Biczak, Varbanescu, & Iosup, 2013; Moniruzzaman & Hossain, 2013).

These databases conform to the CAP theorem, which states that a distributed database can only conform to two of the three desired properties in databases, namely Consistency, Availability and Partition-tolerance. These properties can be interpreted as follows (Gessert et al., 2017):

1. Consistency refers to atomic operations for reads and writes, i.e., every operation is consistent across the database regardless of which client requests the data and regardless of which machine hosts the data.
2. Availability refers to the requirement that all machines in the cluster will always respond to requests for read or write operations and carry out those operations successfully, as long as the particular machine that receives the request is not offline.
3. Partition-tolerance refers to the ability of the database to remain functional despite the loss of some of the machines or a breakdown of communication.

NoSQL databases are generally shared-data systems, since they need to support horizontal scalability in order to accommodate the requirements imposed by Big Data sources (Gupta et al., 2017). The CAP theorem can be interpreted as a simple way of determining whether a NoSQL database conforms to the consistency or the availability requirement in the case of a failure that causes a network partition (Gessert et al., 2017).

2.4 Big Data analytics

A large variety of approaches have been suggested to address the ingestion, processing and storage of Big Data, especially in the case of unstructured social media data. Ghani, Hamid, Targio Hashem, & Ahmed (2019) identifies a number of social media data sources, namely: micro-blogging (such as Twitter), news articles and their comments, blog posts and their comments, internet forums, reviews and Q&A sites. This content can then be subdivided into either text or multi-media content, which in turn can be either images, videos or audio files (Elgendy & Elragal, 2014). This data is generally unstructured and opinionated, which makes it valuable for both businesses and researchers.

The characteristics of the various types of analyses that can be applied to these data sources can then be categorised as descriptive analytics (making use of existing data and reporting on it at face value), diagnostic analysis (performing deeper analysis of the data, such as data mining), predictive analysis (extrapolating from existing data in order to provide decision support) or prescriptive analysis (similar to predictive analysis, but considering every possible outcome and employing meta-analyses such as game theory to provide a more favourable result). Sentiment analysis generally involves diagnostic analysis, as it is a more in-depth analysis of existing data, but it may also be used as part of predictive and prescriptive analytics (Ghani et al., 2019).

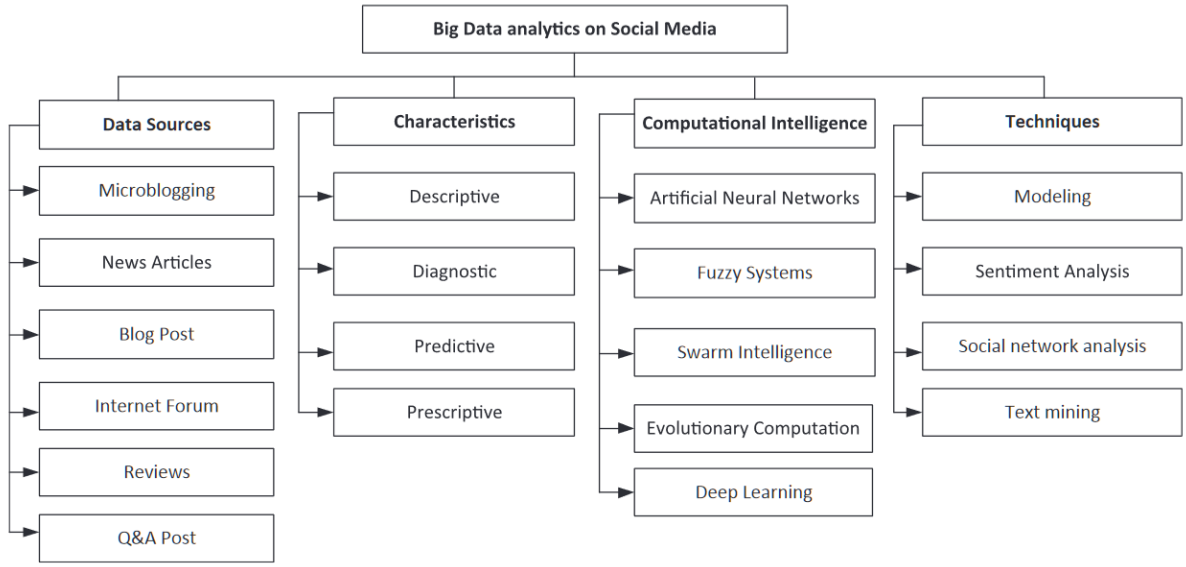

Figure 2.1: An overview of social media Big Data analytics. Adapted from Ghani et al. (2019)

Ghani et al. (2019) then expands some of the possible examples of computational intelligence methods, including artificial Neural Networks, fuzzy systems, swarm intelligence, evolutionary computation and deep learning that can be applied in situations where analysis of social media Big Data is required. Elgendy & Elragal (2014) also discusses MapReduce (a parallel work distribution and combination algorithm) as an underlying mechanism to support such methods in distributed systems.

A variety of techniques can be used to perform the aforementioned analysis as applications of the computational intelligence methods discussed previously. These can include social media modelling, text mining (such as sentiment analysis), social network analysis, advanced data visualisation and visual discovery (Elgendy & Elragal, 2014; Ghani et al., 2019). This study will focus specifically on sentiment analysis as a form of social media Big Data analytics.

2.5 Sentiment Analysis

Sentiment analysis, also known as opinion mining (Pang & Lee, 2008), can provide useful insights into customers' feelings toward businesses and their products (Asur & Huberman, 2010; Fan & Gordon, 2014; Go, Huang, Bhayani, & Twitter, 2009; Bing Liu, 2012; Mäntylä, Graziotin, & Kuutila, 2018; Narr, Hulfenhaus, & Albayrak, 2012; Nodarakis, Tsakalidis, Sioutas, & Tzimas, 2016; L. Zhang, Ghosh, Dekhil, Hsu, & Liu, 2011) and should especially

be considered in the age of Big Data with the advent of social media such as Facebook, Twitter and Tumblr.

Sentiment analysis is a process to determine if a natural language text, or part of it, is subjective or not, and if subjective, whether it expresses a positive or negative view (Taboada, 2016). This process can take place at one of several different levels or contexts.

2.5.1 Document or message level analysis

Document level sentiment analysis assumes that the entire document (or message) expresses a single sentiment on a single entity (Pang & Lee, 2008; Pozzi, Fersini, Messina, & Liu, 2016). This means that the document as a whole receives a classification of positive or negative (or sometimes neutral). This form of sentiment analysis works well with shorter social media messages such as Tweets, since the restrictive message length makes it less likely for more complex opinions involving multiple entities to be present.

2.5.2 Sentence level analysis

Sentence level sentiment analysis is a more refined approach compared to document level sentiment analysis. As the name suggests, this type of sentiment analysis assumes that each sentence presents a single opinion regarding a single entity (Pang & Lee, 2008). This means that for a longer document, each sentence can receive its own classification pertaining to its own content.

2.5.3 Entity and aspect level analysis

Aspect level sentiment analysis is a further refinement of document and sentence level sentiment analysis. Unlike the other two approaches, this technique recognises the sentiments expressed in relation to each entity in a document, which allows for more detailed result. For example, if a longer-form product review refers to multiple other competing products and their attributes, entity level sentiment analysis could distinguish the sentiments expressed and associate each with its corresponding product, and that product's features (Pang & Lee, 2008; Pozzi et al., 2016).

2.6 Sentiment analysis approaches

Several surveys and comparisons have been done to explore the algorithms and techniques used in sentiment analysis literature (Berry, 2004; Bing Liu, 2012; Medhat, Hassan, & Korashy, 2014; Mohey & Hussein, 2016; Pang & Lee, 2008).

Berry (2004) provides a complete overview of text classification techniques as part of a larger text mining review. It includes a theoretical comparison of various algorithms associated with text classification without referring to sentiment analysis and classification directly, but documents a number of the more low-level pre-processing and representation techniques well.

Bing Liu (2012) focuses specifically on sentiment analysis (mostly from a theoretical perspective), and explores the problems associated with the field before describing the approaches and techniques that can be used to address every step of the process. The text also contains a number of more advanced topics such as the use of opinion summarisation and lexicon generation.

At a more practical level, Medhat et al. (2014) performed a literature survey of published articles concerning the application of sentiment analysis, documenting the data source, algorithms used, languages concerned and a number of other aspects of each research paper. This is a particularly useful source, as it provides a good overview of the techniques and data sources studied.

Mohey & Hussein (2016) takes a similar approach, but focuses on the challenges associated with sentiment analysis based on concerns highlighted in literature. This paper includes the techniques used, the accuracies obtained and the dataset used in each case, and also indicates whether the challenges faced in each study is of a theoretical or practical nature.

Pang & Lee (2008) provides what is probably the most complete overview of sentiment analysis, including both theoretical and practical information on the techniques and challenges associated with the field. This survey of the research area is comprehensive in that it includes a wealth of sources from the earliest examples of sentiment analysis up to modern techniques available at the time of writing (2008).

Based on these surveys, sentiment analysis approaches can generally be divided into two major categories: lexicon-based and machine learning (Medhat et al., 2014). These two categories can be subdivided further into more specific approaches, as shown in Figure 2.2.

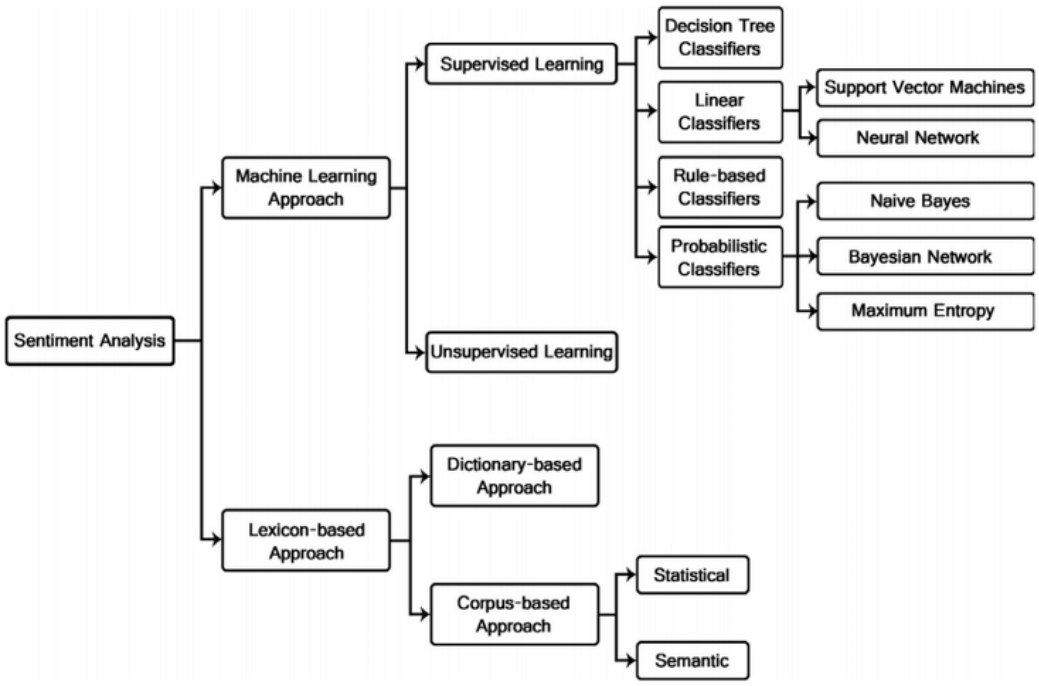

Figure 2.2: Sentiment analysis taxonomy. Adapted from Medhat et al. (2014)

This section will broadly discuss each of these approaches in terms of their technical background, advantages and disadvantages.

2.6.1 Lexicon-based approaches

A lexicon-based sentiment analysis approach is one of the simplest and fastest methods available. It relies on a pre-existing lexicon containing a list of words and associated scores or classifications. Every word in the sample document is then matched to a word in the lexicon, and the scores or classifications are then tallied up to determine an overall classification. This technique, while very fast and easy to use, has a number of shortcomings:

Accuracy: This technique considers every word independently of the words surrounding it, unless an *n*-gram lexicon is used, which contains classified word sequences of *n*-words in length. If *n* is two, then these word sequences can also be referred to as bigrams.

Availability: While there are many lexicons available freely, there are often limitations in terms of the languages covered and the domain it has been trained on. For example, a lexicon created based on the language used in a newspaper will not be effective in a social media domain (Medhat et al., 2014).

Completeness: Lexicons are often incomplete in terms of the vocabulary it covers, and any word that is not contained within a lexicon has the possibility to reduce the accuracy of the classification obtained from it if that word occurs in the sample document Medhat et al., 2014). This can, however, be supplemented by machine learning approaches to improve the term coverage (Khuc et al., 2012).

Lexicon-based sentiment analysis approaches can be categorised into dictionary-based and corpus-based approaches.

2.6.1.1 Dictionary-based

A dictionary-based approach is initiated by assembling a small collection of seed words (a small set of words from which a lexicon can be expanded) which have been manually tagged. This initial collection is then expanded by finding synonyms and antonyms of these words in a larger corpus such as WordNet (Miller, 1995) and adding them as additional seed words to the initial batch. Such an approach can be run iteratively to keep increasing the size of the lexicon. This approach is not domain specific and therefore lacks the necessary context to be applied to specific fields (Kharde & Sonawane, 2016; Medhat et al., 2014).

2.6.1.2 Corpus-based

A corpus-based approach also relies on an initial set of seed words, but unlike the dictionary-based approach, makes use of a large domain-specific corpus. This corpus is used to find other opinion words by identifying cases where the seed words are syntactically connected to other adjectives through conjunctions such as *and*, *or*, and *but*, through which inferences can be drawn to roughly determine the orientation of the other adjective. While imperfect, this approach can be applied in combination with a dictionary-based approach to create a larger, domain-specific lexicon (Hatzivassiloglou & McKeown, 1997; Medhat et al., 2014).

Generally, lexicon-based approaches have the added advantage of being easier to port to a different problem domain or language, rather than re-labelling a new dataset from scratch.

2.6.2 Machine learning approaches

Machine learning is an area of artificial intelligence that allows a program to learn and improve from a set of examples (usually called a training set) without having to explicitly define the set of rules that will lead to the desired outcome. Machine learning approaches generally make use of large input datasets to train digital models in order to best fit them for use in a certain domain. These approaches can be grouped into supervised, semi-supervised and unsupervised categories.

2.6.2.1 Supervised machine learning

Supervised machine learning approaches make use of a labelled input dataset which indicates the correct classification for each document. This guides the model towards a desired output for a particular set of inputs, thereby supervising the training process (Medhat et al., 2014). This section will discuss a number of examples of supervised learning in literature, specifically pertaining to three approaches: Naïve-Bayesian classifiers, Support Vector Machines and Neural Networks.

2.6.2.1.1 Naïve-Bayesian

Multinomial Naïve-Bayesian classifiers apply Bayesian statistics to determine the probability that a classification applies to a sample. The technique is considered naïve due to the fact that it assumes all features (properties of the item being classified) contribute independently to the likelihood that an input belongs to a particular class, regardless of any actual correlations between the features (Medhat et al., 2014). The multinomial aspect refers to the assumption that the features will follow a multinomial distribution and is recommended for discrete count values (e.g. which could be expected from a count vectoriser), but which is often also applied to decimal values generated by a term frequency-inverse document frequency (TF-IDF) vectorizer. See section 5.5.1 for more detail on vectorizers, including TF-IDF.

Naïve-Bayesian classifiers rely on the probability that a document may belong to a class based on similar documents previously belonging to the same class. This probability can be expressed as follows (Gamallo & Garcia, 2014):

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \quad (1)$$

Where *d* denotes a document from a domain of documents:

$$\mathbb{D} = \{d_1, d_2, \ldots, d_n\}$$

And *c* denotes a class from a set of *m* possible classes:

$$\mathbb{C} = \{c_1, c_2, \ldots, c_m\}$$

With a collection of unique words:

$$\mathbb{W} = \{w_1, w_2, \ldots, w_s\}$$

Where each $w \in \mathbb{W}$ occurs in at least one $d \in \mathbb{D}$. Note that the denominator $P(d)$ remains constant, since it does not depend on *c*. It is assumed that each word *w* occurs independently of every other word in a document *d* for a class *c*. This simple model of assumed independence is the reason for the classifier's naïvety (Gamallo & Garcia, 2014).

Naïve-Bayesian sentiment analysis can generally be expected to have good performance (as long as the number of classes is kept low) and is easy to implement. It has one major disadvantage for sentiment analysis purposes: the assumption that features are independent does not reflect the reality of the complex interactions and relationships between features in natural language (Medhat et al., 2014).

A number of existing research papers have been published on the use of Naïve-Bayesian classifiers in a variety of contexts. Troussas, Virvou, Espinosa, Llaguno, & Caro (2013) investigated the use of such a classifier for sentiment analysis of Facebook statuses. This study is interesting since its investigation of Facebook statuses (user-generated posts) involves the analysis of much longer documents than the common approach of using Twitter data (5 000 characters versus 140 characters). They made use of a total of roughly 7 000 statuses, split

evenly between positive and negative classes. They found that their classifier had a precision of 0.77, recall of 0.68 and an F-measure of 0.72. This was a favourable performance in comparison to the Perceptron classifier that they also tested (0.65, 0.56 and 0.60 for precision, recall and F-measure, respectively), but performed worse overall in comparison to a Rocchio classifier (0.75, 0.73 and 0.74 for precision, recall and F-measure, respectively).

An investigation into a scalable Naïve-Bayesian classifier was performed by Liu, Blasch, Chen, Shen, & Chen (2013). They created a Hadoop MapReduce implementation of a Naïve-Bayesian classifier and trained it on a large movie review dataset. The seven-node Hadoop cluster was trained and tested incrementally to examine the impact that a larger dataset has on accuracy. Their results are summarised in Figure 2.3 and Figure 2.4:

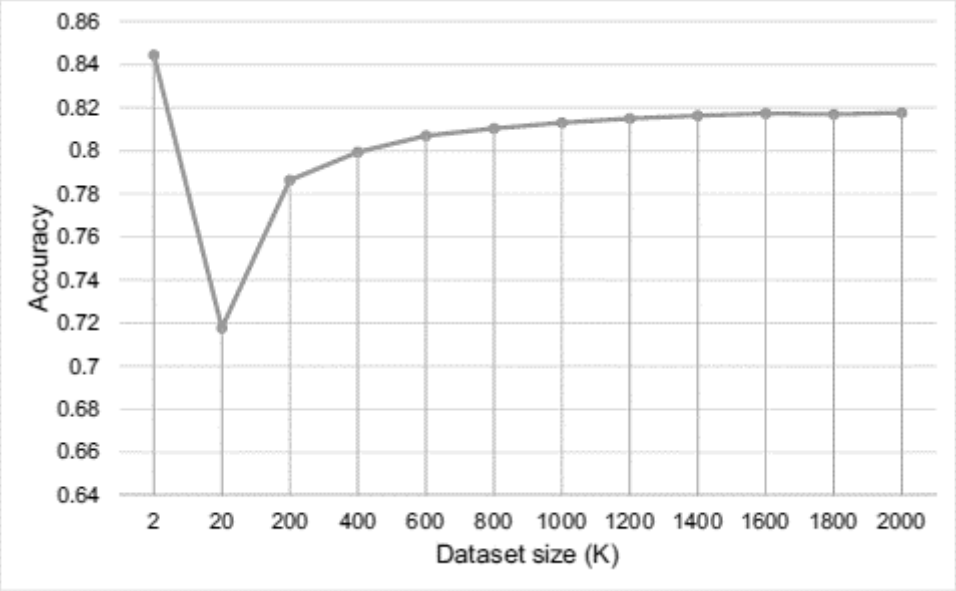

Figure 2.3: Naïve-Bayesian accuracy versus training set size. Adapted from Liu et al. (2013)

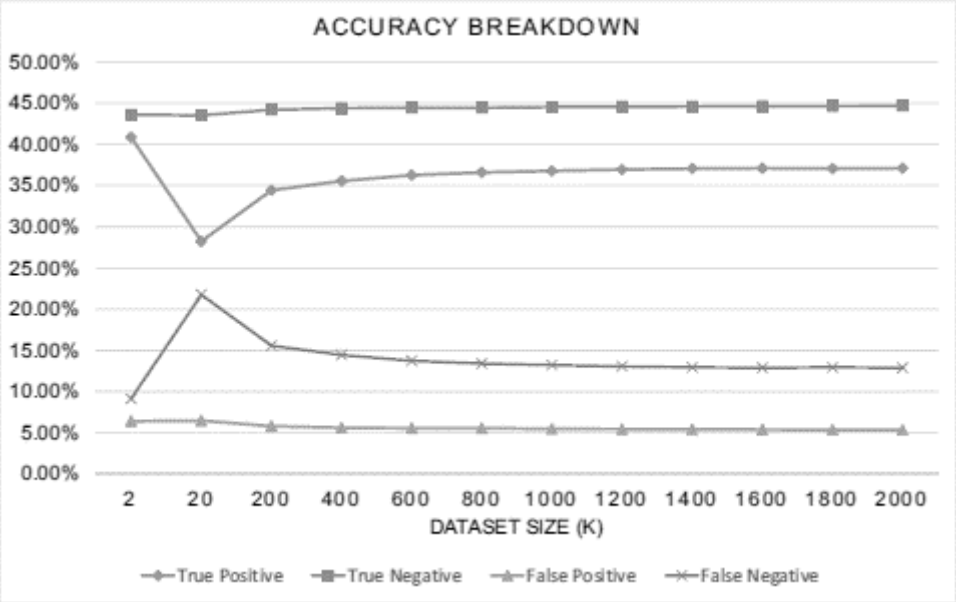

Figure 2.4: Naïve-Bayesian accuracy breakdown versus training set size. Adapted from Liu et al. (2013)

Based on their results shown in Figure 2.3 and Figure 2.4, it is clear that the accuracy improves dramatically with the introduction of more data, up to around one million samples, from where the accuracy levels off and the difference gained for every two hundred thousand samples diminishes.

These examples from literature show that the Naïve-Bayesian approach can yield good results in a variety of situations, and can be applied effectively to large datasets.

2.6.2.1.2 Support Vector Machine

Support Vector Machines model documents as vectors in *n*-dimensional space, and then attempts to find a hyperplane, which is represented by a vector, separating the categories with the widest possible margin (Mohri, Rostamizadeh, & Talwalkar, 2013). Once the model has been trained and the optimal hyperplane has been found – see Figure 2.5 right – classification can be performed by determining on which side of the hyperplane a document's vector falls (Pang, Lee, & Vaithyanathan, 2002).

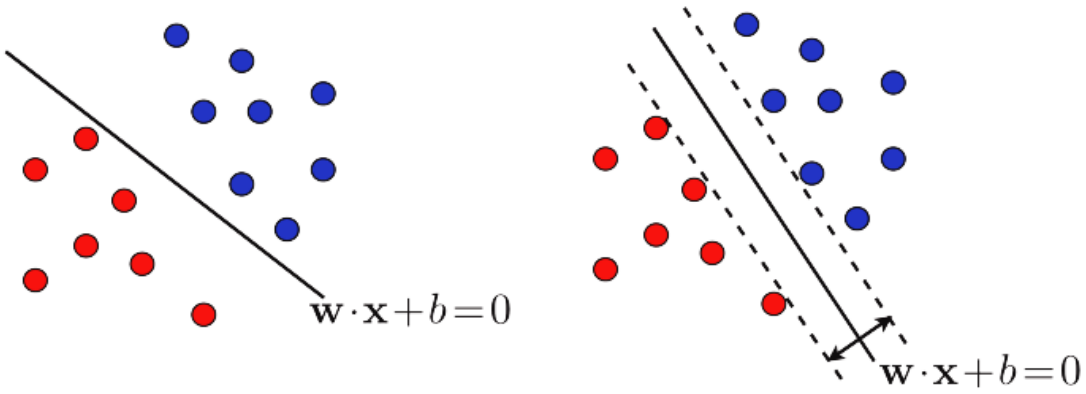

Figure 2.5: Two representations of a hyperplane separating two categories or classes. Adapted from Mohri et al. (2013).

Vapnik (2000) describes a support vector machine as a mapping of a set of "... input vectors x into a high-dimensional feature space $\mathbb{Z}$ through some nonlinear mapping, chosen *a priori*." The mapping is the process of representing the input data in the form of a vector that can be represented in n-dimensional space (a concept commonly referred to as vectorization, performed by a vectorizer). In the case of sentiment analysis, this input data is usually in the form of natural language text that is then converted using vectorizers that make use of algorithms such as term frequency-inverse document frequency (TF-IDF).

Vapnik (2000) presents two problems that need to be solved for Support Vector Machines:

1. The conceptual problem of finding a separating hyperplane that generalises well.
2. The technical problem of how one should treat high-dimensional spaces computationally.

Theorem 5.2 proposed by Vapnik (2000) reads as follows:

Theorem 5.2

If training sets containing ℓ examples are separated by the maximal margin hyperplanes, then the expectation (over training sets) of the probability of test error is bounded by the expectation of the minimum of three values: the ratio m / ℓ, where m is the number of support vectors, the ratio $[R^2|w|^2]/\ell$, where R is the radius of the sphere containing the data and $|w|^2$ is the value of the margin, and the ratio n /ℓ, where n is the dimensionality of the input space: $EP_{error} \leq E\left\{\min\left(\frac{m}{\ell}, \frac{[R^2|w|^2]}{\ell}, \frac{n}{\ell}\right)\right\}$

Theorem 5.2, therefore, justifies the generalisation capacity for model in high-dimensional space (Burges, 1998; Vapnik, 2000) and therefore solves the conceptual problem. The technical problem of representing the high-dimensionality computationally is solvable since the capability of calculating the inner products between the support vectors and the feature space vectors does not require the explicit construction of the feature space (Vapnik, 2000).

The hyperplane separating the classes will usually have $n - 1$ dimensions in n-dimensional space. This is visible in Figure 2.5, where the vector space is two-dimensional and the hyperplane separating them is a one-dimensional line. In three-dimensional space, the hyperplane may be a two-dimensional surface.

SVMs are generally reported to have higher accuracy than other Naïve-Bayesian classifiers (Joachims, 1998). Many implementations of SVMs are generally available as open source packages. This makes the technique quick and easy to apply for users who may not have the capacity or time to implement it themselves. An SVM can generally cover a domain as well as the training material allows. As with any machine learning technique, it is necessary to ensure that the training domain and the application domain is as similar as possible for maximum accuracy. Many SVM implementations make use of the LibSVM library (Chang et al., 2011), including Scikit-learn (Pedregosa et al., 2011), which is used in the case of this research.

Tripathy, Agrawal, & Rath (2016) investigated a number of sentiment analysis approaches, including a Support Vector Machine, and compared the accuracy between them based on the

use of any combination of unigrams, bigrams and trigrams. Their methodology included a pre-processing step of removing stop words, numbers and special characters and then generating a sparse matrix based on the output from a vectoriser. Four models (Support Vector Machine, Naïve-Bayesian, Maximum Entropy and Stochastic Gradient Descent) were then trained with these matrices as input. Their results indicated that the Support Vector Machine performed optimally when a combination of unigrams, bigrams and trigrams were used, with a maximum accuracy of 88.94%. This combination of classifier and features provided the best results, with other classifiers achieving 86.23% (Naïve-Bayesian), 88.48% (Maximum Entropy) and 85.11% (Stochastic Gradient Descent) accuracy.

Varghese & Jayasree (2013) reported the use of a Support Vector Machine for aspect-based sentiment analysis (i.e. sentiment analysis pertaining to specific attributes or characteristics of items). They made use of 2 500 digital camera reviews from a variety of online stores. This dataset contained, on average, around two hundred reviews per product. The subjective sentences were selected using the SentiWordNet lexicon (Esuli, 2006), and the absence of any opinion words in a sentence meant that it was discarded. This filtering was done to reduce the processing overheads during training. The Stanford Deterministic Coreference Resolution system was also employed to ensure that opinions are matched at the sentence level (i.e., to make sure that the opinion relating to a pronoun of a preceding or following sentence is matched up with the nouns in the sentence under evaluation. Finally, adjectives from the SentiWordNet lexicon were then manually labelled in conjunction with specific aspects (e.g. battery life or camera size) during training to preserve contextual information. Based on this series of pre-processing steps, it was possible to generate charts such as the one shown in Figure 2.6 for a particular camera model.

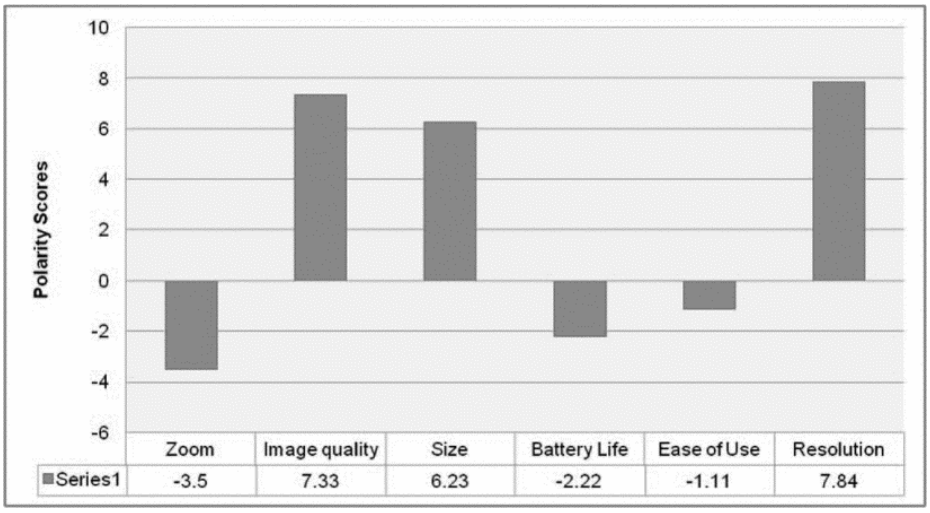

Figure 2.6: Sample output from an aspect-based Support Vector Machine. Adapted from Varghese & Jayasree (2013)

This system achieved an overall accuracy of 77.98%. The complete evaluation results are shown in Table 2.1below:

Table 2.1: Aspect-based Support Vector Machine accuracy (Varghese & Jayasree, 2013)

Product	**Precision**	**Recall**	**Accuracy**
Camera 1	85.94%	87.30%	78.48%
Camera 2	93.30%	77.38%	74.82%
Camera 3	91.30%	84.00%	80.65%
Average	90.18	82.89	77.98

These results are promising since they illustrate a successful combination of various techniques in conjunction with a Support Vector Machine to achieve fairly accurate aspect-based predictions.

2.6.2.1.3 Neural Network

Artificial Neural Networks (ANNs) are based on the suspected structure of biological Neural Networks and are represented as a series of neurons organised in layers. All references to the term "Neural Network" in this study implies the use of an ANN. The connections between the neurons are weighted, influencing the input that each neuron in the next layer receives. If the combined input exceeds the activation threshold, the neuron is activated, and its output is passed to the next layer (Raschka, 2015).

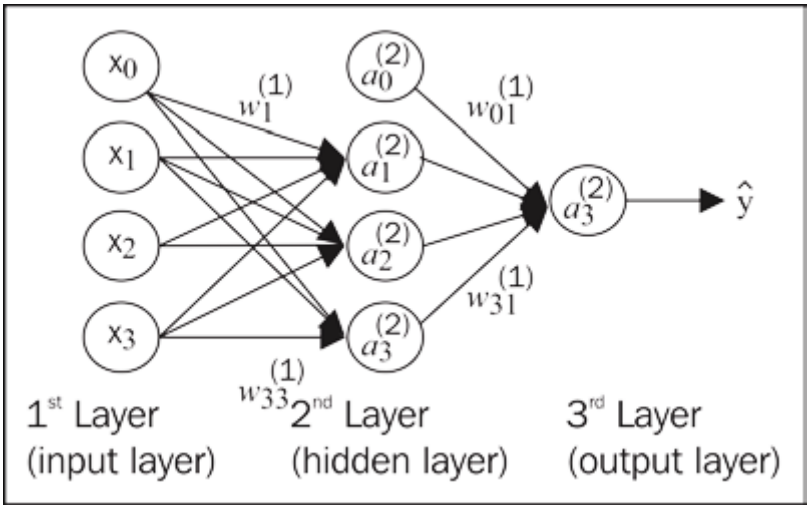

Figure 2.7: A general representation of a Neural Network showing the neurons in the input layer, hidden layer and output layer. Adapted from Raschka (2015)

The weights of the connections are determined by training the network with a set of inputs and a desired set of outputs. A generalised graphical representation of such a network is shown in Figure 2.7. The weights are continually adjusted according to the training data in an attempt to fit the network to optimally solve the classification problem (Raschka, 2015).

Haykin (2004) provides an in-depth overview of the types of Neural Networks, along with their characteristics. There are a number of different architectures in which the components of a Neural Network can be arranged and can interact. The three fundamental architectures identified by Haykin (2004) are: single-layer feed-forward Neural Networks, multi-layer feed-forward Neural Networks and recurrent networks.

A single-layer feed-forward network only consists of an input layer and an output layer. This kind of network, therefore, has no hidden layers. A graphical representation of such a network is shown in Figure 2.8.

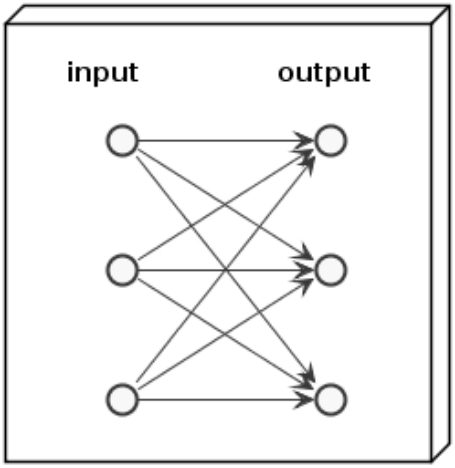

Figure 2.8: A graphical representation of a single-layer feed-forward Neural Network based on Haykin (2004)

The addition of one or more hidden layers allows for additional processing of the inputs to be completed prior to reaching the output layer. Such layers can allow Neural Networks to perform more complex functions, and eventually leads to the creation of deep learning (discussed later on in this section) through the addition of many layers (Haykin, 2004). A graphical representation of a Neural Network with one hidden layer in addition to the output layer is shown in Figure 2.9.

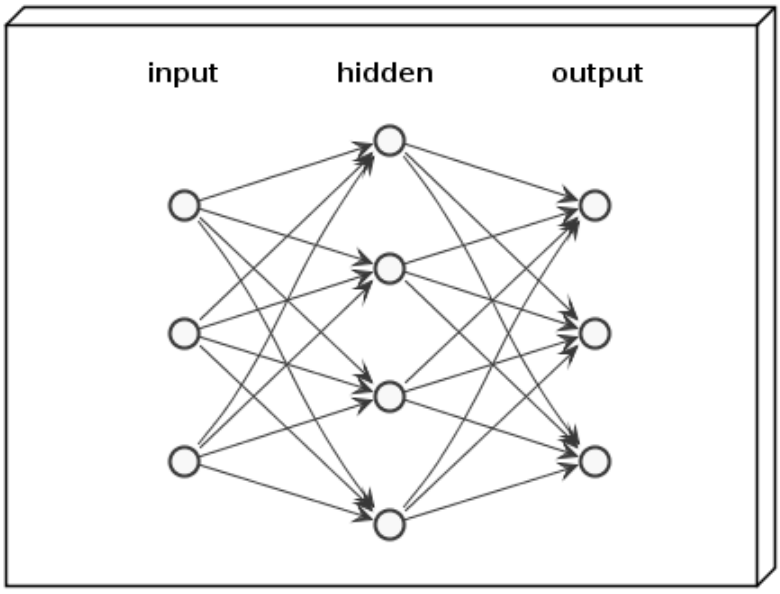

Figure 2.9: A graphical representation of a multi-layer feed-forward Neural Network based on Haykin (2004)

These two examples of Neural Networks exclusively allow information to pass from the input layer, through any intermediate layers, to the output layer, in other words, the information can only pass through the neurons in one direction. If the possibility exists for information to pass backwards from the output layer the Neural Network is classified as recurrent (Haykin, 2004). A simple recurrent Neural Network without any hidden layers is shown in Figure 2.10.

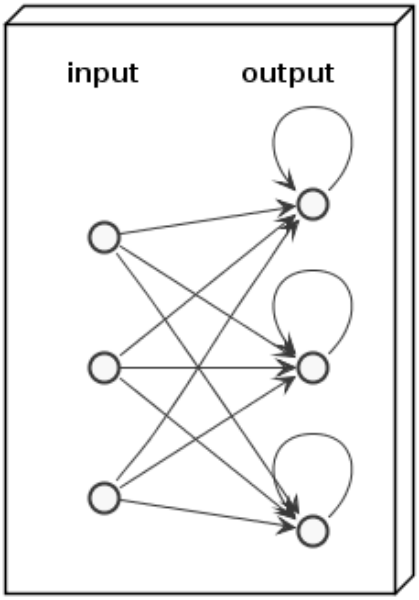

Figure 2.10: A graphical representation of a recurrent Neural Network based on Haykin (2004)

The addition of a feedback loop in the Neural Network increases both the learning capability of the network and the resources required to run it (Haykin, 2004). The passage of information from the output layer towards the hidden and input layers is known as backpropagation.

For the optimisation of Neural Networks, it is necessary to make several architectural decisions in terms of the number of input nodes and what they represent, the number of hidden layers (and the number of neurons for each), and how the output is calculated. Each neuron can have an activation function associated with it, and these functions are generally one of the following (Duch & Jankowski, 1997):

1. A linear function which passes the weighted input it received to the output without performing any processing or calculation on it.
2. A threshold function, which activates when a value above a certain threshold is received and then passes a high value to the output.

3. A step function, which is a form of threshold function which outputs a value of 1 if the input exceeds the threshold, and a -1 if it does not.
4. A sigmoid function, which is a generalised version of a step function that takes the shape of a sigmoid.

The weights of the connections are calculated during training to provide the optimal set of outputs given a specific set of inputs. Initial weights are usually assigned random values and then changes are applied to the weights by some predefined value denoted by Δ. The training sample can be denoted by *E*, such that:

$$\Delta = \eta\big(t(E) - o(E)\big)x_i \tag{2}$$

Where η denotes the constant learning rate, $t(E)$ denotes the target value for sample E, and $o(E)$ denotes the observed value for E. The weight w_i for the i-th input unit x_i is therefore changed by Δ after every sample. It should be noted that a smaller η will result in a slower learning rate that would require more iterations to converge to the correct result, but a larger value for η may result in a sub-optimal weight. The value for η is a hyper-parameter that should be chosen beforehand and tested iteratively until an optimal value has been determined for a specific case.

There are a number of examples in literature where Neural Networks were applied for sentiment analysis purposes. Duncan & Zhang (2015) made use of a feed-forward Neural Network to perform sentiment analysis on Twitter data. Prior to training, they executed a number of pre-processing steps to remove punctuation, mentions, single characters and stop words. They then ran a word stemming algorithm to reduce the overall feature space. They used a small training dataset of only two hundred Tweets, which were split equally into positive and negative classes. A further one hundred Tweets were then used for testing, from which they obtained an overall accuracy of 74.15%. Precision and recall were not stated. They reported memory issues if more than two hundred Tweets were used during the training steps, and that an attempt to reduce the feature space further resulted in a major drop in accuracy down to only 31.04%.

Sharma & Dey (2012) explored the possibility of using a backpropagation Neural Network for binary classification of movie and hotel reviews. Their proposed approach achieved an

accuracy level of 95% on the movie reviews dataset (compared to between 80% and 89% for the three lexicons they also tested). A chart of their results as a function of feature set size is shown in Figure 2.11 below:

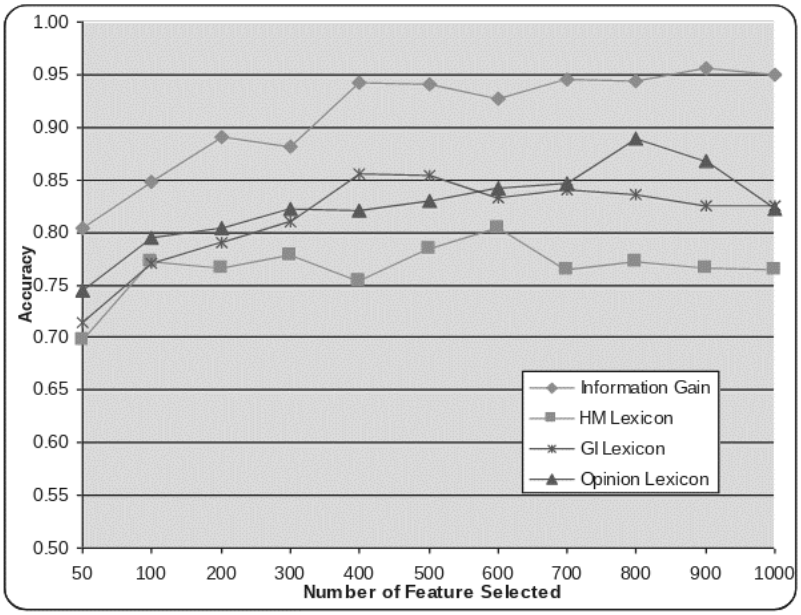

Figure 2.11: Neural network accuracy versus lexicons on movie review dataset. Adapted from Sharma & Dey (2012)

Their findings show a general increase in accuracy obtained by the Neural Network as the number of features are increased. This trend continues with the use of the hotel reviews dataset as well, as shown in Figure 2.12 below:

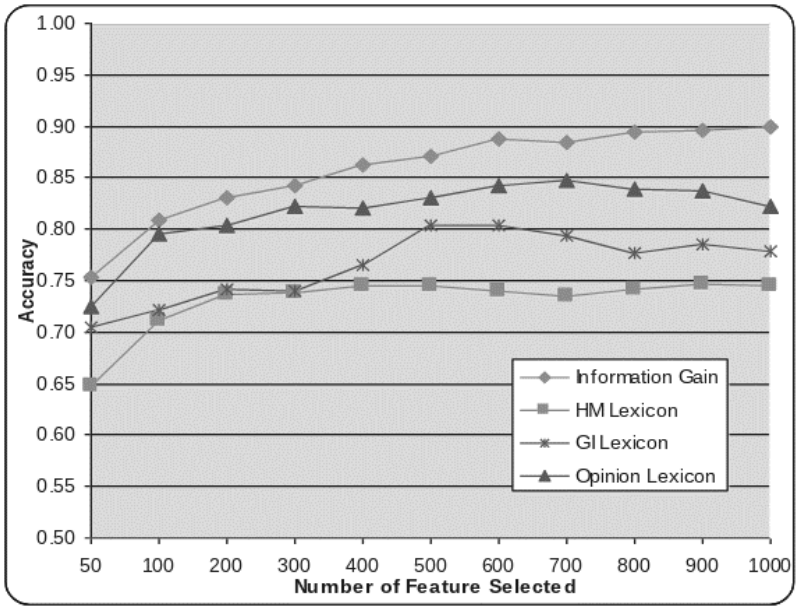

Figure 2.12: Neural network accuracy versus lexicons on hotel review dataset. Adapted from Sharma & Dey (2012)

Their results show that the back-propagation Neural Network exceeds the capabilities of the lexicons at every feature count, on both datasets.

2.6.2.2 Semi-supervised machine learning

In cases where domain-specific, labelled training data is not available, it is possible to make use of generalised training data (Medhat et al., 2014) with high confidence classifications to serve as pseudo-labelled training data. This was applied successfully by He & Zhou (2011) to achieve acceptable classification performance in two different datasets.

Sindhwani & Melville (2008) proposed a semi-supervised sentiment analysis algorithm that infers sentiment based on lexical training with unlabelled samples. Their study made use of two sets of blogs (one set related to enterprise software and another related to presidential candidates in the United States) and a sample of movie reviews, in order to test a variety of domains. Their tests included a comparison between a number of different classifiers, including their proposed Semi-supervised Lexical Regularised Least Squares (SS+LEX+RLS) classifier. They found that their classifier exceeded the accuracy of other classifiers (a lexical classifier,

a lexical Regularised Least Squares classifier, a linear Support Vector machine and a Transductive Support Vector Machine) tested in almost all cases, and obtained accuracy measures of between 60% and just over 90% depending on the dataset tested.

Ortigosa-Hernández et al. (2012) proposed a multi-dimensional sentiment analysis classifier to detect subjectivity, sentiment polarity and will-to-influence using a semi-supervised approach. Their proposal suggested that existing multi-dimensional Bayesian network classifiers be adapted to make use of a modified expectation maximisation algorithm. They found that their approach performed favourably in a multi-dimensional situation compared to other one-dimensional approaches and that an increase in the amount of data may be useful to improve accuracy overall. They did, however, express some concern regarding the ability of their proposed approach to scale well with much larger datasets, as this has not been tested.

2.6.2.3 Unsupervised machine learning

Unsupervised techniques are used in cases where labelled training data is not available or difficult to obtain, and are generally applied when a large amount of unlabelled data is available. These techniques tend to rely on information such as sentence similarity, lexical association or semantic orientation. Purely unsupervised learning is difficult to apply effectively, and semi-supervised techniques tend to be applied in similar situations where it is possible to provide at least some minimal form of weak supervision from which the training process can extrapolate (Medhat et al., 2014).

Hu, Tang, Gao, & Liu (2013) proposed the use of emotional signals such as emoticons and product ratings as a means of creating an unsupervised sentiment analysis classifier. Their expectation is that an emoticon or rating represents a post-level (document-level) emotion signal, whereas a sentiment lexicon identifies word-level emotion signals. They set up a list of positive and negative emoticons, as shown in Table 2.2 below:

Table 2.2: Emoticons for the use of unsupervised learning (Hu et al., 2013)

Emotion	**Emoticons**
Positive	:) :) :-) :D =)
Negative	:(: (:-(

They then identified two Twitter datasets to test whether a classifier based on the expectation that a positive emotion signal would accompany a positive post would provide favourable results. Their classifier was based on an orthogonal nonnegative matrix tri-factorisation model that assumes that features can be clustered based on their distribution in the samples. They found that their proposed approach performed better than other baseline approaches (Lexicon-based, MoodLens and Constrained nonnegative matrix factorisation).

Fernández-Gavilanes, Álvarez-López, Juncal-Martínez, Costa-Montenegro, & Javier González-Castaño (2016) investigated the use of dependencies between lemmatised tagged words as part of a sentiment propagation approach. They found that their method gave them an improvement of between three and seven percent over existing methods due to its novel use of linguistic meaning with regard to intensification, adversative and concessive relations, negation and modification.

2.6.2.4 Evaluation metrics

The accuracy of a technique is determined by the number of true positives (*tp* – samples correctly classified as positive), false positives (*fp* – samples incorrectly classified as positive), true negatives (*tn* – samples correctly classified as negative) and false negatives (*fn* – samples incorrectly classified as negative) classified by the algorithm. These factors can be combined into two metrics known as precision and recall, and are calculated as follows:

$$precision = \frac{tp}{tp + fp} \tag{3}$$

$$recall = \frac{tn}{tn + fn} \tag{4}$$

Essentially, precision and recall represent the fractions of correct positive predictions and correct negative predictions, respectively. These two elements can be combined into a single metric known as the F-measure (or F-score). The relationship between the F-measure and precision and recall variables is shown in the equation below:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5}$$

Which has been derived from the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{6}$$

The equation above is also known as the F_1 measure, which assigns equal weights to precision and recall. The $F_{0.5}$ measure weighs precision higher than recall, and the F_2 measure weighs precision lower than recall.

Accuracy can also be given as a simple percentage of the number of times the algorithm correctly classified a document out of the total number of documents. This determination of accuracy happens to be the sum of the number of true positives and true negatives, divided by the sum of the true and false positives, and true and false negatives. This relationship can be expressed as follows, and subsequently simplified:

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn} \tag{7}$$

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions\ made} \tag{8}$$

This provides a simple percentage of correct predictions in cases where the relationship between precision and recall is not crucial.

2.6.2.5 Multi-class and multi-label classification

The majority of research on sentiment analysis makes use of only binary classification, i.e., a system of classifying texts into two classes, usually “positive” and “negative”. Some research classifiers add a third class for “neutral” texts. These classifiers are referred to as ternary classifiers (Bouazizi & Ohtsuki, 2016).

Some research has been done into multi-class classification, such as the work published by Bouazizi & Ohtsuki (2016). Their research investigated the possibility of classifying Tweets into seven different classes using a pattern-based approach. They compared accuracy figures between binary, ternary and multi-class classification using their approach, and found that their

binary classifier (with positive and negative classes) achieved an accuracy of 87.51%, whereas their ternary classifier (with the addition of the neutral class) only achieved an accuracy of 83.0% - a significant reduction. The accuracy was further reduced with the application of the multi-class model (which made use of seven classes: happiness, love, neutral, sadness, anger, hate and sarcasm) to 56.9% overall.

A study by Lin, Yang, & Chen (2007) investigated multi-class classification of news articles based on the emotions evoked in the readers of the articles. The dataset was based on a feature of the Chinese Yahoo! news website that prompted readers for their feedback on each article. Readers were given eight choices: happy, angry, sad, surprised, heart-warming, awesome, bored, and useful. Since "useful" is generic feedback rather than an indication of emotion, the authors performed their experiments twice, with one experiment including the "useful" class, and the other excluding it. Their study made use of the LibSVM classifier (Chang et al., 2011), which they used to set up a variety of features with different weights. Accuracy varied widely between the classes, with their final measurements shown in Table 2.3 below:

Table 2.3: Multi-class classification accuracy obtained by Lin, Yang, & Chen (2007)

Class	**Accuracy**
Awesome	62.84%
Heart-warming	62.40%
Surprised	59.10%
Sad	64.67%
Useful	89.66%
Happy	77.44%
Bored	79.87%
Angry	73.89%

Note that their final results included the use of the "useful" class. The range of results they obtained is striking, with a difference of almost 30% between the "Surprised" and "Useful" classes.

Generally, studies of this nature are rare in comparison to the number of studies in which binary classifiers are used. Binary classifiers also tend to be more flexible in the sense that all text/document classification techniques can accommodate at least two classes, but the options for multiple classes are more restricted.

A multi-label classification approach assigns more than one label to each instance. The argument is generally that certain labels may have interrelated meanings, and that combinations of labels may provide more accurate and meaningful results. Huang, Peng, Li, & Lee (2013) note that there are generally two approaches to multi-label classification: class reorganisation (also called problem transformation (S. M. Liu & Chen, 2015)) and algorithm innovation.

Class re-organisation involves the transformation of multi-label classification into, what is effectively, single-label classification by selecting a single label from the list randomly, ignoring multi-label instances or creating single labels based on the combinations of multi-label instances. This approach is generally not effective since it either discards some of the multi-label information, or in the case of combinations, some combined labels may end up with too few instances to be useful. Algorithm innovation, on the other hand, involves the modification of existing classification models to support multi-label approaches instead (Huang et al., 2013). An example of this is the multi-label k-nearest neighbour (ML-KNN) algorithm proposed by Zhang & Zhou (2007) as an adaptation of the existing k-nearest neighbour algorithm which can handle multiple labels.

Liu & Chen (2015) conducted a study comparing various multi-label classification methods based on their accuracy using two Twitter datasets. Their comparison included eight state of the art methods including: binary relevance (Boutell, Luo, Shen, & Brown, 2004), classifier chains and classifier chain ensembles (Read, Pfahringer, Holmes, & Frank, 2011), HOMER (Tsoumakas, Katakis, & Vlahavas, 2008), RAKEL (Tsoumakas, Katakis, & Vlahavas, 2011), ML-KNN (M.-L. Zhang & Zhou, 2007), calibrated label ranking (Fürnkranz, Hüllermeier, Loza Mencía, & Brinker, 2008) and random forests of predictive clustering trees (Kocev, Vens, Struyf, & Džeroski, 2007). They found that, for one dataset, RAKEL and classifier chain ensembles performed the best, while calibrated label ranking, HOMER and classifier chain ensembles performed the best on the other dataset.

Huang, Peng, Li, & Lee (2013) investigated the use of a multi-task, multi-label classification approach, whereby Tweets were given multiple labels such as “Complaint, Care/Support”. This approach allows one to quickly, for example, identify problem areas, since the complaint label will be accompanied by another label indicating the topic of the complaint. Another set of labels might be “Positive, Promotion”, indicating that a Twitter user has had a positive

experience regarding a promoted product or service. Their research made use of three sentiment classes (positive, negative and neutral) and a large set of topic classes (promotion, compliment, news and complaint, among others). They found that their multi-task, multi-label model resulted in an increase of 5% on sentiment classification and 12% on topic classification compared to other approaches.

This study will make use of binary classifiers, since the training data derived from the data collected only provides two classes: positive and negative sentiment. This also makes the resulting metrics more widely applicable, as binary classifiers with binary labels are the most common classifiers in research.

2.6.2.6 Ensemble classifiers

It is also possible to apply a combination of classifiers simultaneously and then determine a consensus-based result established on the majority of classifications. This technique is commonly called ensemble classification (Neethu & Rajasree, 2013). Xia, Zong, & Li (2011) made use of an ensemble technique that integrates multiple classifiers with the aim of increasing accuracy. They tested Naïve-Bayesian, Maximum Entropy and Support Vector Machine classifiers, in conjunction with part-of-speech and word-relation feature sets. The ensemble methods applied were fixed combination, weighted combination and meta-classifier combination.

Their study employed multiple datasets, including the Cornell movie-review corpuses and the Multi-Domain Sentiment Dataset, which includes product reviews of different types of products from the Amazon.com online store. In terms of part-of-speech feature sets (with multiple strategies), they found that the highest average (across all product types) accuracy achieved by a single classifier was that of the Maximum Entropy classifier at 86.10% on the movie review dataset. The greatest accuracy achieved by the ensemble classifier was 86.60% on the movie review dataset. They concluded that an ensemble of feature sets and classification approaches perform better than a combination of feature sets alone. Their word-relation feature sets (which made use of unigrams, bigrams and dependencies) performed best under all circumstances when all three feature sets were used (as opposed to any individual one of the three). They concluded that a combination of both feature sets and classification approaches

performed better compared to either ensembles of only feature sets or only classification approaches (Xia et al., 2011).

2.6.2.7 Other considerations

It is important to note that, unlike lexicon-based approaches, machine-learning approaches cannot easily be ported to other domains or languages (Taboada, Brooke, Tofiloski, Voll, & Stede, 2011). Such a change would require retraining the classifier, and would therefore need new training data, which needs to be labelled for supervised learning approaches.

2.6.3 Hybrid approaches

Hybrid approaches where combinations of lexicon-based and machine learning classifiers are applied have been documented in literature (Medhat et al., 2014). It has been found in some cases that a combination of classifiers can provide improved results. For example, in the case of Martín-Valdivia, Martínez-Cámara, Perea-Ortega, & Ureña-López (2013), a combination of supervised and unsupervised approaches in the form of multiple machine learning classifiers and the SentiWordNet (Esuli, 2006) sentiment lexicon was used to create a classifier that exceeded the performance of any individual classifier tested in their scenario.

The following section will provide an overview of the current applications of sentiment analysis, indicating the need for high-performance algorithms that are applicable to social media.

2.7 Applications of sentiment analysis

Early detection and response towards customer unrest can prevent widespread negative publicity, as mentioned by Fan & Gordon (2014) in three different cases. Both the Red Cross and Burger King posted messages on social media that received negative responses from the community but defused their respective situations by swiftly responding publicly and admitting their mistakes. Chapstick had also been implicated in a similar situation, but responded by removing customers' comments from their website, thereby worsening their public image in their attempts at censoring customers' opinions. Early detection of unrest and attempts at damage control on social media are becoming increasingly common, as noted by a number of authors (Chen et al., 2012; Fan & Gordon, 2014; Minelli et al., 2013).

There are even fears of malicious customer manipulation by companies in order to improve their public image (Pang & Lee, 2008). The use of sentiment analysis has also been shown to be useful in predicting future trends. Asur & Huberman (2010) showed that Twitter data combined with sentiment analysis can be used to predict movie box office revenues before release in a way that outperformed the predictions of the Hollywood Stock Exchange. They concluded that social media presents a collective wisdom that could provide a useful avenue for accurate prediction of future outcomes. Making use of this potential source of new insights requires sentiment analysis software that can easily be distributed across multiple machines. Some of the currently available services and software packages are discussed in the following section.

2.8 Twitter sentiment analysis software

There are a number of commercial software packages and services (Lexalytics, 2015; SAS, 2015; Sentimeter, 2015) that offer sentiment analysis as part of their product offering, but generally they do not directly discuss the algorithms (or combination thereof) that they make use of, presumably for competitive reasons. There are, however, some free and open source libraries available that offer a wide variety of text classification algorithms, including the Stanford NLP library (The Stanford NLP Group, 2015), which makes use of deep learning (recursive Neural Networks) and the Natural Language Toolkit (Bird, Klein, & Loper, 2009) which includes a Naïve-Bayes classifier.

In literature, there are numerous examples of document classification algorithms being used to perform sentiment analysis, which is essentially document classification (Roebuck, 2012). For example, Khuc et al. (2012) made use of a purely lexicon-based classifier and compared it with a lexicon-and-learning-based classifier. Lexicon-based classifiers are very common since they are fast and easy to implement (Kouloumpis et al., 2011; Pang & Lee, 2008; Taboada et al., 2011). Many studies also make use of Naïve-Bayes, Support Vector Machines and Maximum Entropy machine learning techniques (Agarwal et al., 2011; Go, Bhayani, & Huang, 2009; Narr et al., 2012; Pang & Lee, 2008; Wang, Can, Kazemzadeh, Bar, & Narayanan, 2012). Several studies made use of artificial Neural Networks to perform sentiment analysis (Ghiassi, Skinner, & Zimbra, 2013).

2.9 Twitter sentiment analysis approaches

Various studies have been done to refine sentiment analysis techniques in attempts to increase their accuracy. This section will discuss the various approaches found in literature.

Sentiment analysis has generally been applied at three different levels in literature: document-level, sentence-level and entity-level. Twitter sentiment analysis (TSA) is generally applied using document-level classification. This means that the sentiment of the document as a whole is considered, which assumes that each document (in this case each Tweet) only contains one particular viewpoint or opinion. In the case of very short documents such as Tweets, the length dictates that the number of differing opinions that can be expressed will be low. Document-level classification is, therefore, a good match for social media platforms, such as Twitter, which encourage or enforce shorter submissions.

2.9.1 Lexicon-based TSA approaches

A number of Twitter-specific lexicons have been developed to increase the accuracy of classifying Tweets. Lexicons developed specifically are useful since they are tailor-made to accommodate the various domain-specific characteristics of Tweets, such as hashtags (topics of discussion or emotion, e.g. #happy), emoticons (which can be graphic Unicode symbols, or a combination of standard punctuation characters) and reTweets (sharing a Tweet, frequently denoted by an RT-prefix).

Lexicon-based approaches are also attractive for Tweet classification since they can be applied directly without requiring any training beforehand, as long as the document language being classified corresponds to the language for which the lexicon was created. This considerably lowers the barrier to entry for large-scale Tweet classification, and can provide results that are generally sufficient to divide a customer base roughly into satisfied and unsatisfied customers.

Lexicons are, however, at a disadvantage compared to machine learning approaches when it comes to the variable nature of social media platforms such as Twitter, where nomenclature and meaning can change rapidly according to social influence.

Mohammad, Svetlana, & Zhu (2013) built sentiment lexicons from Twitter data due to the fact that the accuracy of sentiment lexicons and algorithms depend on the domains where they are trained and used. Ideally, the training domain should match the test domain as closely as possible to ensure the maximum accuracy. This ensures that the classifier is trained to recognise the jargon used in the applicable domain, e.g. Twitter users make use of hashtags and emoticons which are not present in movie reviews. This means that a classifier trained in movie reviews will perform poorly when applied to Twitter data. The classifier created by Mohammad et al. (2013) owed the most significant portion of its accuracy to their own sentiment lexicons, largely overshadowing the influence of part-of-speech (POS) tagging and the removal of emoticons and hashtags.

2.9.2 Machine learning TSA approaches

Machine learning approaches to Twitter sentiment analysis can be applied effectively due to the large quantity of data available publicly, and free of charge in most cases. For supervised machine learning the minimum requirement for labelling could be the existing use of emoticons in Tweets, from which the sentiment of the rest of the document can be inferred and labelled automatically. Tweets could also be labelled manually through human evaluation, and the brief nature of Tweets means that they can be labelled quickly. Having such a large amount of data available is attractive for supervised machine-learning purposes, since a large, ever-expanding pool of data allows for an iterative learning process of labelling, training, validation, and testing on dataset sizes previously unattainable prior to the advent of social media.

Pak & Paroubek (2010) collected Twitter data from the Twitter API based on a series of emoticons to automatically set up a corpus of positive and negative Tweets (based on the emoticons used). They tested a Naïve-Bayes classifier, along with SVM and CRF classifiers, but found that the Naïve-Bayes classifier provided the greatest results. The Naïve-Bayes classifier was trained on two datasets, one making use of n-gram presence as a binary feature (instead of using a count vector) and the other making use of part-of-speech tags. They achieved the best performance when making use of bigrams, and that this performance could be further improved by accounting for negation. They also found that the accuracy of the classifiers could be improved by increasing the number of samples, although this is only effective up to a certain point, as shown in Figure 2.13 below:

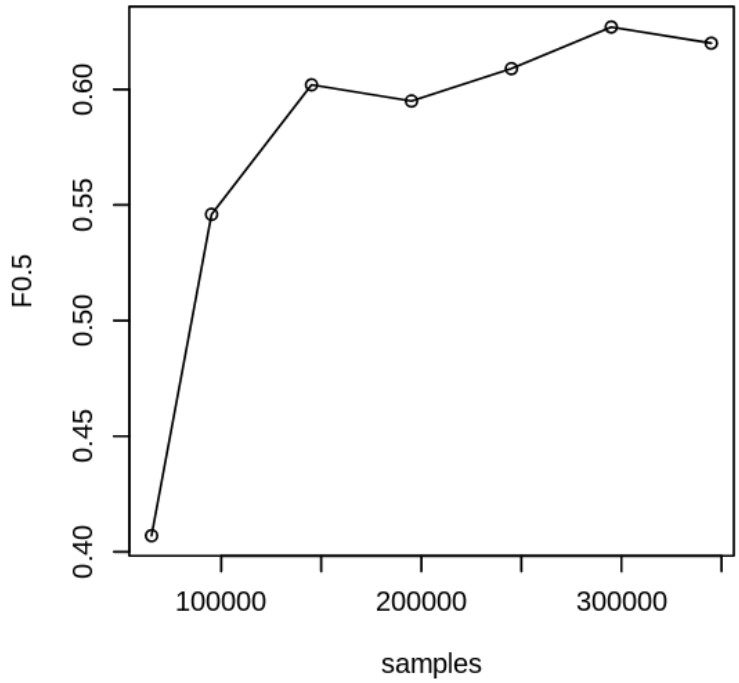

Figure 2.13: $F_{0.5}$-measure as a function of sample size. Adapted from Pak & Paroubek (2010)

Gamallo & Garcia (2014) made use of a Naïve-Bayesian classifier to detect the polarity of English Twitter data. They used a combination of two Twitter datasets totalling around 12 000 Tweets in order to train two different classifiers: a baseline classifier that used an unmodified training corpus, and a binary classifier that was trained on a modified corpus that was simplified to remove neutral Tweets (based on the lack of words found in a polarity lexicon). They found that the binary classifier outperformed the baseline classifier in all comparative tests.

2.9.3 Hybrid TSA approaches

A combination of lexicon-based and machine-learning classifiers can be applied to Twitter sentiment analysis. This strategy has been documented in literature by a number of authors:

Khuc et al. (2012) applied a combination of a lexicon-based approach and an adaptive logistics regression technique to determine a sentiment score for Tweets in which the words used are not found in the lexicon. This approach achieved an increase in classification accuracy and better coverage of terms than a lexicon can independently provide. Their study also documented the generation of a Twitter lexicon using only emoticons as seed words.

Ghiassi et al. (2013) made use of a machine learning approach to select and reduce features for use in a lexicon-based classifier. They reported a significant reduction in the feature set and a simplified data model with an increase in classification accuracy.

2.10 Popularity in literature

A number of Scopus (Scopus, 2020) queries were performed to determine the most popular sentiment analysis techniques in literature. A similar approach was taken by Mäntylä, Graziotin, & Kuutila (2018) due to Scopus's broad coverage of academic literature. The searches were all performed on the 1st of July 2016. The results of these queries are given in Table 2.4, with the chosen techniques shaded.

Table 2.4 Summary of Scopus search results for popular sentiment analysis techniques

Technique	**Scopus Query**	**Number of Results**
Lexicon	(TITLE-ABS-KEY(sentiment analysis)) AND (lexicon)	1546
SVM	(TITLE-ABS-KEY(sentiment analysis)) AND ((svm) OR (Support Vector Machine))	1103
Naïve-Bayesian	(TITLE-ABS-KEY(sentiment analysis)) AND ((bayesian) OR (bayes))	740
Neural network	(TITLE-ABS-KEY(sentiment analysis)) AND (Neural Network)	649
K-means	(TITLE-ABS-KEY(sentiment analysis)) AND (k means)	541
Decision tree	(TITLE-ABS-KEY(sentiment analysis)) AND (decision tree)	256
Bag-of-words	(TITLE-ABS-KEY(sentiment analysis)) AND (bag of words)	173
Latent semantic indexing	(TITLE-ABS-KEY(sentiment analysis)) AND (latent semantic indexing)	147
K-nearest neighbour	(TITLE-ABS-KEY(sentiment analysis)) AND (k nearest neighbour)	100
TF-IDF	(TITLE-ABS-KEY(sentiment analysis)) AND ((tf-idf) OR (term frequency inverse document frequency))	74
Random forest	(TITLE-ABS-KEY(sentiment analysis)) AND (random forest)	74
Expectation maximisation	(TITLE-ABS-KEY(sentiment analysis)) AND (expectation maximization)	11
Total	**(TITLE-ABS-KEY(sentiment analysis))**	**6555**

This study will, therefore, make use of a lexicon-based system, Multinomial Naïve-Bayes, a linear Support Vector Machine and an Artificial Neural Network due to their popularity in literature. The machine learning algorithms will be provided by Scikit-learn (Pedregosa et al.,

2011) and the lexicon will be provided by AFINN (Nielsen, 2011). Both implementations are available as open source Python libraries.

2.11 Related work

Stewart & Singer (2012) investigated the performance difference between a single machine multithreaded system (the Java/C/Haskell versions of Fork/Join) and a cluster system (Hadoop Map/Reduce). It was found that the single machine performed far better than the cluster for smaller jobs, but performed very poorly when the input sizes grew larger than a few megabytes. The Hadoop cluster was over four times faster than the C implementation of Fork/Join when faced with a dataset of roughly 100 megabytes. Their conclusion was to propose a hybrid system that makes use of both multithreading and distributed computing to gain maximum performance. From this, it is important to note that in order for a sentiment analysis cluster to be useful, the input data stream must be sufficiently large to ensure that the added network latency is negligible when compared to the performance gains of multiple machines. This study will differ in that the algorithms compared will all be parallelised and distributed, as opposed to only be parallelised (as in the case of Fork/Join). This follows their concluding proposal to make use of both multithreading and distributed computing.

Khuc et al. (2012) discuss a large-scale sentiment analysis system that makes use of a MapReduce framework and a distributed database model (Apache Hadoop and HBase) and found that the performance of such a system is not proportional to the size of the cluster. Instead, while there was a performance increase, it was less pronounced with each additional machine. This study will include a similar analysis showing the performance impact of each machine added to the cluster in order to illustrate this phenomenon and allow for the extrapolation of such data. This study will differ in that it will compare many different algorithms in terms of an increased number of metrics.

Ha, Back, & Ahn (2015) reported the use of a Hadoop cluster for distributed sentiment analysis. They discussed the use of one master node, one slave node and two data nodes, where only the two data nodes are responsible for loading and analysing the data. The overall architecture was given as shown in Figure 2.14:

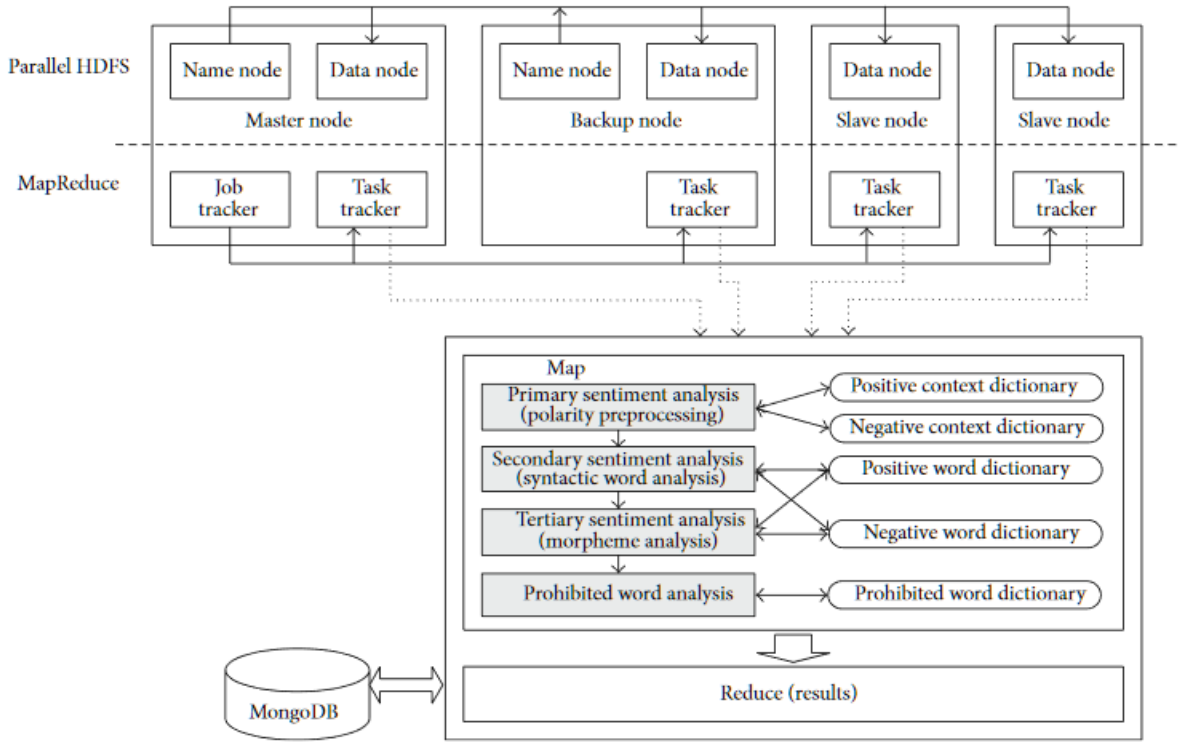

Figure 2.14: A Hadoop-based distributed sentiment analysis architecture. Adapted from Ha et al. (2015)

This study is useful for Hadoop-related investigations since it reports concrete resource usage figures, such as the actual amounts of memory, CPU load and time required for each experiment, and how it was measured across all four machines. Unfortunately, there was no comparison between one and two data nodes, as two data nodes were used for all the experiments. This study will differ in that it will consider multiple different sentiment analysis approaches, and report on the ways in which the metrics are affected by the addition of more machines to the cluster.

Nodarakis, Tsakalidis, Sioutas, & Tzimas (2016) reported on a distributed sentiment analysis cluster running on Apache Spark. They made use of hashtags and emoticons for automatic annotation of training and testing data, and proposed a novel algorithm for sentiment classification that is tailor-made for the Spark environment. They describe the speedup and scalability of their algorithm as more machines are added to the cluster, but do not provide memory and processing usage figures. Their comparison of running time for classification using binary and multi-class hashtags and emoticons is given in Figure 2.15 below:

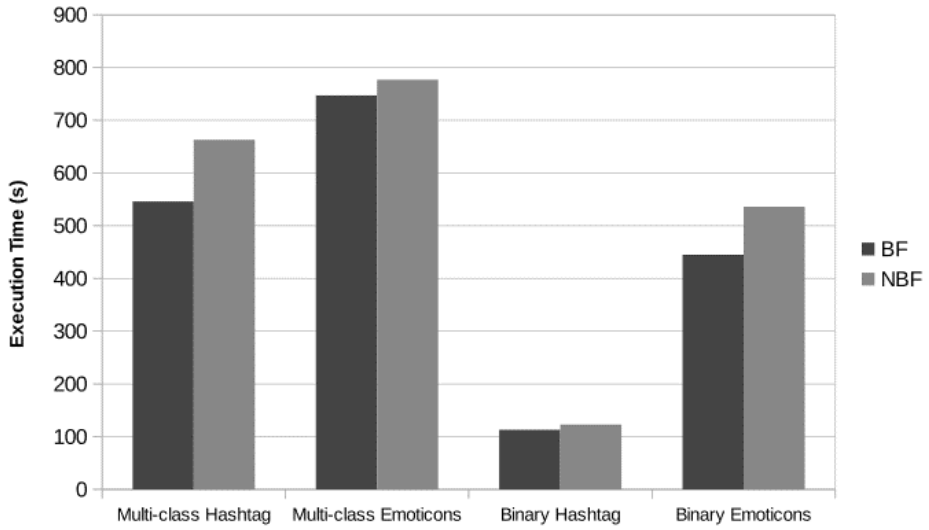

Figure 2.15: Reported sentiment analysis execution time on a distributed Apache Spark cluster. Adapted from Nodarakis et al. (2016)

This report of execution time is not ideal, as it considers the entire test set as a group instead of reporting on the average per-sample execution time. Having an average value of time required to classify a single sample is more useful as it allows one to more easily extrapolate the time that would be required for datasets of different sizes. This study will report a per-sample execution time, and will include multiple sentiment analysis approaches.

2.12 Gap analysis

As stated in Chapter 1, the first research objective of this study (RO1) is to determine the current gap in literature as a means of informing the rest of this study. In order to do so, we can summarise the factors reported in the most relevant literature explored in the previous sections. These factors and the corresponding papers are given in Table 2.5 below.

Table 2.5 Summary research papers most closely related to the aims of this study

	Factors investigated				
Research paper	**Accuracy**	**Resource usage**	**Multiple machines**	**Multiple cluster sizes**	**Multiple approaches**
Stewart & Singer (2012)		X	X	X	
Khuc et al. (2012)	X		X	X	
Ha et al. (2015)		X	X		
Nodarakis et al. (2016)	X		X		X

Two conclusions can be drawn from Table 2.5. Firstly, the number of papers investigating similar combinations of factors are very few, and secondly, none of the existing research papers discussed cover the complete set of factors proposed in this study.

It is therefore possible to conclude that, generally, literature investigating the overall performance of sentiment analysis approaches in distributed environments is sparse. The gap in literature therefore lies in the lack of a combined study that includes all of the factors mentioned in the studies discussed so far, including reports of accuracy (with accuracy percentages, precision, recall, and F-measure) and resource usage (memory, processing and time) across a variety of cluster sizes and multiple sentiment analysis approaches. Based on the above literature, this study improves on existing research by collecting a much wider variety of metrics, of an increased number of sentiment analysis techniques, in a parallelised and distributed environment.

2.13 Summary

This chapter investigated the characteristics of Big Data, and the challenges they pose in terms of ingestion, storage and analysis. A discussion of NoSQL as a primary solution to the storage aspect of Big Data followed, which included the various categories available in conjunction with the strengths and weaknesses of each. This provided the background which informed the storage back-end which was utilised by this study, which will be discussed in more detail in Chapter 3. This was followed by a discussion of Big Data analytics in general, in order to place the remainder of the chapter into context. Sentiment analysis as a type of Big Data analysis

was then introduced as the focus of this study. The various sentiment analysis approaches documented in literature was investigated, and the four most popular variants for examination in this study was identified. It was shown that the current research into these sentiment analysis algorithms largely focusses on accuracy alone and rarely discusses the computational requirements to achieve those numbers.

Additionally, such experiments have usually been performed on stand-alone computers, instead of multiple computers working in parallel to maximise performance. This limits the usefulness of these studies since they do not provide any insight into the possible performance benefits of parallelisation or the impact of network communication and load balancing involved with such systems. This lack of holistic investigation into the performance characteristics of sentiment analysis approaches under real-world conditions was identified as the gap in current literature. The investigation in this chapter informed the choice of four approaches identified for further evaluation in this study, based on a review of a popularity ranking of these algorithms in literature. The approaches chosen were: lexicon-based, artificial Neural Networks, Support Vector Machines and Naïve-Bayesian analysis. This background formed the literature study of the dissertation, and served to inform the decisions made regarding the choice of research methodology, the research environment, and the experimental design and methodology, discussed in Chapters 3, 4 and 5, respectively.

The following chapter will, therefore, proceed to evaluate the possible research paradigms which would support an investigation of this nature. Based on the paradigm chosen, a research methodology will be chosen as basis for the research methods required, followed by a discussion of possible research instruments. This will place the decision to make use of a formal experiment design into the broader context of the research, and inform the research process which will be followed going forward. The chapter will conclude with a discussion of the analysis and interpretation of the data to be performed, and a review of the limitations posed on the study by the chosen research methodology. .

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Introduction

Chapter 2 discussed the characteristics of Big Data and the challenges they pose in terms of ingestion, storage and analysis. This was followed by an investigation of NoSQL as a possible distributed storage solution. The various categories, with their strengths and weaknesses, were discussed to place the decision made in this chapter into context. A discussion of Big Data analytics followed, and sentiment analysis as a subset thereof was investigated in detail. The current approaches to sentiment analysis research and comparison, as well as gaps in literature, were discussed. Four sentiment analysis approaches were chosen for investigation in this study based on their popularity in literature and serves as the background for the research approach subsequently followed in this study.

This chapter will present possible research methodologies and explain the reasoning for the chosen methodology, in order to inform the research process going forward. This will, firstly, rely on a review of the various research paradigms available, to form the basis of this study, as illustrated by the methodological pyramid. Following which a research methodology will be identified, which will, in turn, inform the research methods most appropriate given the context provided by the preceding two chapters. Based on the aims of this study and the nature of the data which will be generated, a positivist research paradigm will be chosen, in conjunction with a formal experiment as the research instrument. The chapter will conclude with an overview of the research process, in the context of the structure of this dissertation, as followed, along with a discussion of the limitations of this study imposed as a consequence of the choice of research methodology.

3.2 Research Paradigms

Research paradigms represent the base of the methodological pyramid, shown in Figure 3.1 below. The research paradigms form the foundation of the research approach and informs the research methodologies which apply. The data collection methods then derive from these methodologies.

Figure 3.1: The methodological pyramid. Adapted from Zikmund, Babin, Carr, & Griffin (2010)

Research paradigms represent shared schools of thought and approaches to study phenomena. This section will briefly discuss three major paradigms – interpretivism, positivism and pragmatism - in the context of Information Systems (IS) research, and whether they fall into qualitative or quantitative categories (Oates, 2005).

3.2.1 Qualitative

Qualitative research deals with subjects where precise measurements and objective calculations are not feasible, such as social interactions, perceptions or interpretations. Interpretivism is a major qualitative paradigm that is commonly found in IS research.

3.2.1.1 Interpretivism

Interpretivism is focused on studies pertaining to social situations and attempts to understand the way people perceive the world. This paradigm does not make use of hypotheses, and accepts the world as being subjective according to each individual's experience. This means that it does not require or even encourage a single objective truth, instead it attempts to explain shared understandings among people in social situations through many possible interpretations (Oates, 2005).

This approach works well in situations where societal views and experiences need to be investigated to understand the reasons behind the decisions that people make when they interact in situations of interest to IS research. This can, for example, be a study into why the software

development team at one company has a different view on what the correct way is to perform certain tasks or use a specific workflow than another company with similar requirements. In such an environment, an interpretivist study would attempt to understand the situation through a series of social interactions (such as interviews and surveys) to reach a series of plausible, convincing explanations (Oates, 2005).

As this study does not include a social aspect, it does not make use of an interpretivist paradigm.

3.2.2 Quantitative

Quantitative research is concerned with definite, measurable observations and statistical analysis in order to determine objective truths that lead to laws that can universally be applied. Positivism is a common research paradigm applied to quantitative studies in IS research.

3.2.2.1 Positivism

Positivism forms the basis of the scientific method. It assumes that the world is ordered, it behaves in a deterministic, non-random manner and can be investigated objectively. This assumption of order and determinism underlies the ability to find patterns which can be used to explain phenomena. The second assumption that the world can be investigated objectively is based on an assumption that these deterministic patterns function independently of the personal thoughts and experiences of individuals and can be examined without individual bias (Oates, 2005).

The application of positivism in terms of the scientific method relies on three techniques known as reductionism, repeatability and refutation. Reductionism involves scaling down complex systems into individual components which can more easily be studied and tested independently. This makes it more straightforward to determine cause and effect since the variables involved are reduced to the bare minimum. Repeatability is an expectation that multiple experiments should yield the same result under the same conditions. This is to ensure that the results of a single experiment were not accidental or variable. Repeatability ensures that the results obtained are reliable and that reasonable conclusions can be drawn from them. Refutation allows other researchers to challenge the results of another study if it does not correspond with their own results under the same or similar circumstances. This can call into question the validity and repeatability of the claims made in the original research (Oates, 2005).

It may be difficult to apply reductionism in all cases, especially in complex, real-world situations where it can be problematic to account for every variable or to remove the complexity from the environment (Oates, 2005).

3.2.3 Mixed methods

Mixed methods make use of a combination of qualitative and quantitative methods in an attempt to provide a practical alternative to the two worldviews discussed, a sort of grey area between the black and white options of positivism and interpretivism. This mixed method takes the form of pragmatism.

3.2.3.1 Pragmatism

Pragmatism prescribes the existence of multiple realities (in opposition to the positivist view that there is only one truth) that may be subject to empirical investigation as an approach to solving problems in the real world. This allows the researcher to perform their research unconfined by the two opposing systems of positivism and interpretivism by combining a mixture of quantitative and qualitative methods in order to provide the most complete results possible given the ambiguities and uncertainties often involved in research (Feilzer, 2010).

This study follows a purely positivism-guided research paradigm, since it is the most suitable for quantitative, comparative research involving concrete figures and measurements.

3.3 Research design

Different kinds of research designs exist to address the variety of questions that arise in studies. There are two broad types of research studies: empirical studies, and non-empirical studies (Mouton, 2001).

Empirical studies typically make use of either primary data from experiments, surveys and case studies or involve the analysis of existing data. Existing data can be divided into text data, which includes content analysis and historical studies, among others, and numerical data, which is concerned with secondary analysis of existing data and statistical modelling. These techniques are quantitative in nature (Mouton, 2001).

Non-empirical studies are generally more philosophical in nature, dealing with conceptual subjects, worldviews and theoretical questions. These studies involve broader topics than empirical studies by making use of a less rigid, qualitative research approach (Mouton, 2001).

This study involves an empirical comparison of techniques concerned with numeric data, which makes it a good fit for an experiment. The following sections will discuss the various types of experiments and the design chosen for this study.

3.4 Research instruments

There are various research instruments available for the collection of data. These instruments can provide either qualitative or quantitative data, or a combination of both.

3.4.1 Questionnaires

Questionnaires are prepared surveys consisting of a set of questions that can be distributed to research participants to complete. Generally, questions are either open-ended or closed-ended. Open-ended questions allow the participant to provide broader information that the researcher may not have anticipated, and is useful in cases where longer-form answers are desired. Closed-ended questions tend to either require yes or no answers, or the selection of one or more answers from a set of choices (Oates, 2005).

This variation in possibilities allows a researcher to collect a combination of qualitative and quantitative data using a single research instrument, depending on the needs of the study. Questionnaires can also be a quick way of collecting a large amount of data from a pool of participants in a short period of time. The pre-prepared structure of the questionnaire also allows the researcher to guide the participant's answers to follow a certain structure to maximise the applicability of the answers to the research project (Oates, 2005).

3.4.2 Interviews

Interviews are one-on-one, in-person conversations between the researcher and the research participant. Like questionnaires, interviews can consist of a combination of open-ended and closed-ended questions, depending on the needs of the study. Unlike questionnaires, interviews

allow the researcher to ask follow-up questions to clarify certain matters or ask the participant to expand on their answers to ensure that all the necessary information is collected (Oates, 2005).

The interviewer (who is usually the researcher) will make notes during the interview, and optionally record the conversations for transcriptions afterwards. This can considerably increase the amount of work involved with interviews, and generally limits the number of participants the research project can accommodate in a given time period (unlike questionnaires which can have very high numbers of participants with little effort), but can ensure that the quality of information collected is of higher quality. Interviews generally generate qualitative data (Oates, 2005).

3.4.3 Experiments

An experiment is an attempt to investigate the cause and effect between the circumstances of a controlled environment and the observed outcome. There are multiple types of experiments with varying degrees of scientific rigour, which can be applied depending on the factors being investigated and the generality of the results required. A highly controlled environment, for example, can yield a definite proof of cause and effect, but may not reflect real-world conditions and might therefore not be applied generally outside of the artificial test environment (Oates, 2005).

Experiments primarily rely on hypotheses. These are predictions of possible outcomes that are tested empirically when an experiment is carried out under controlled conditions. The conditions and outcomes of the experiment are called the independent and dependent variables, respectively. The independent variables are carefully manipulated throughout the experiment in order to observe the resulting changes in the dependent variables in order to determine cause and effect (Oates, 2005).

There are generally two broad categories of experimental designs: formal experiments (also known as laboratory experiments) and field experiments (also known as natural experiments). The difference between them lies in the degree of control that the researchers can exert over the environment in which the experiment takes place (Mouton, 2001; Oates, 2005).

A study performed in a natural environment that is representative of the setting in which the subject is commonly found is considered a field experiment. While such experiments carry the possibility that uncontrollable, external factors may influence the outcome, it does ensure that the results are more generally applicable to real-world environments (Mouton, 2001; Oates, 2005).

A formal experiment, on the other hand, takes place in a laboratory environment under artificial conditions in order to allow for greater control of the factors that may influence the results. This experimental design is only possible in situations where the subject can reliably be isolated from external interference, such as scientific laboratories (Mouton, 2001; Oates, 2005).

Since this study takes place in a controlled computing environment that is free from external interference (i.e., a non-shared environment where all resources will be devoted exclusively to the software running the experiment and the supporting infrastructure such as the Operating System), a formal experimental design can be followed.

3.5 Formal Experiment Design

A formal experiment is a research design that aims to test hypotheses in a controlled laboratory situation. It usually involves a number of measurements and observations of the interaction between dependent and independent variables to determine the cause and effect in a particular situation. This allows a study to reliably explain a situation and make predictions of results based on the values of the independent variables (Oates, 2005).

Carrying out a formal experiment requires control of all the variables in the experiment in order to change only one variable between experiments to determine the factor that influences the results. This can be difficult to achieve in complex situations that have many variables that cannot easily be accounted for. Additionally, such a controlled environment usually represents an artificial environment that does not reflect the real-world and the results obtained cannot be directly applied in real-world situations (Oates, 2005).

While such a high level of control is difficult to achieve, it does mean that a formal experiment is the only research methodology that can *prove* a causal relationship between variables. Any

other research methodology can only show a correlation between variables and present a reasonable explanation for the results obtained (Oates, 2005).

3.6 Research process

This study followed a standard research process as outlined graphically by Oates (2005) in Figure 3.2.

The motivation and problem statement, along with the research questions were discussed in Chapter 1. Chapter 2 documented the literature review that provided the background for the study and reinforced the need for this study to address the current gap in literature. This chapter explained the reasoning for the use of an experiment to address the research questions, and the next chapter will discuss the research environment in which this experiment took place.

The design of the experiment will be discussed in detail in Chapter 5. The data collected from Twitter consisted of a random sample of Tweets identified as English by Twitter. This sample was also filtered server-side to make sure that all Tweets contain either a smiling or frowning emoticon, as this was used to automatically tag the data as positive or negative for training purposes.

From the collection onwards, the observation data was processed through a pipeline of steps (discussed in more detail in section 5.5) consisting of training, validation and testing. The data collected from each of these steps (where applicable) included empirical measurements of resource usage in addition to standard accuracy measurements. This data was then summarised in aggregate and compared in Chapter 6 . Chapter 7 will conclude with the discussion of the research questions in the context of the results obtained, along with future research possibilities and the limitations of this study. This chapter will also include the substantive, scientific and methodological reflection on the completed study.

The data used for training and testing, as well as the classifier output was stored in a distributed NoSQL database, as this allows the data to be distributed across all the machines participating in the experiment at any given time. Section 3.6.1 will discuss the database chosen for this task.

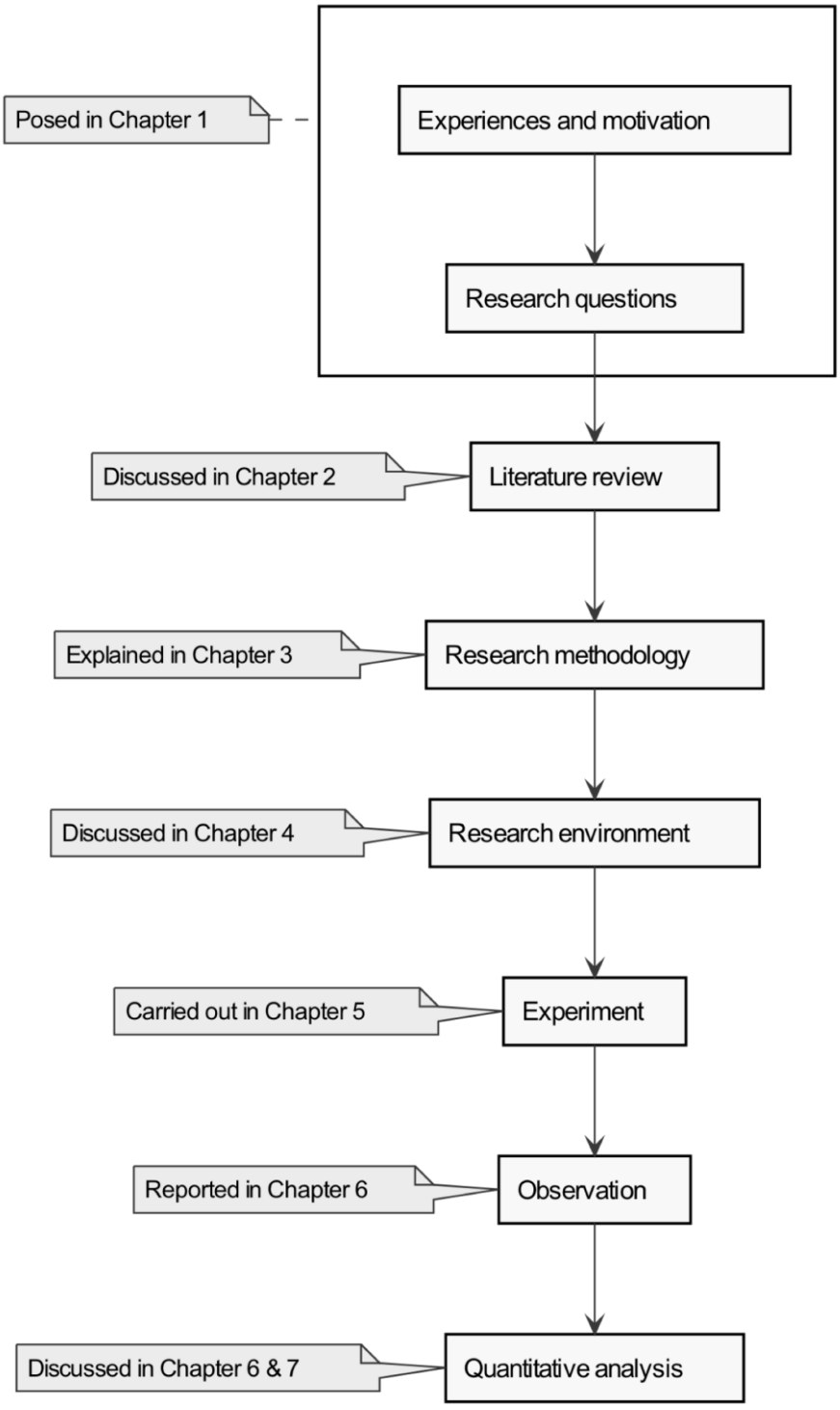

Figure 3.2: The research process followed in this study, adapted from Oates (2005)

3.6.1 Chosen database

This study makes use of a wide-column database, namely: Apache Cassandra (Datastax, 2018a). The motivation for choosing a wide-column database comes down to a number of reasons (Van der Linde & Kotzé, 2018):

1. Scalability: The partitioned data model allows for horizontal scalability across multiple computers on a network while maintaining the ability to support more complex queries than those allowed by key-value stores.
2. Performance: By foregoing features commonly found in relational databases, wide-column databases can provide better performance while still supporting the data modelling necessary for social media data storage.
3. Purpose: This study does not need to store large documents and does not require any graph-traversal operations as part of its experiment. This means a wide-column database is the most suitable for such a study. This may not be the case for all social media data storage, as a study aimed at social network analysis may be better suited to a graph database.

Cassandra was chosen for its prevalence among NoSQL and wide-column databases (Solid IT, 2019), its corporate support, and its active development (Van der Linde & Kotzé, 2018). Section 3.6.1.1 will discuss the features of Apache Cassandra.

3.6.1.1 Apache Cassandra

Apache Cassandra is a distributed wide-column NoSQL database. Cassandra has near linear scalability, high availability and supports petabyte data volumes. It automatically balances data between all the machines in the cluster, and also rebalances whenever a new machine is added, which allows for easy scalability. Replication is built-in and can be customised on a per-table basis (Datastax, 2018a).

Cassandra makes use of a master-less architecture, where every machine in the cluster has the ability to process read and write operations. When a write request is received, it coordinates that operation with other machines to complete the operation. This means that reads and writes scale well, unlike databases which make use of a single master, which can often be a bottleneck for write operations (Datastax, 2018b).

It is important to note that Cassandra does not recognise relationships between tables, and does not support features generally associated with relational databases, such as JOINs or foreign key constraints. This makes it unsuitable for the use-cases generally associated with relational databases.

3.7 Analysis and interpretation of data

All data collected will be summarised using the Python Pandas data science library as a post-processing step. This summarised data will be reported directly for each sentiment analysis approach. The summarised data will then be aggregated again for a further comparison between each approach across the various cluster sizes.

The classifier performance metrics for accuracy as a percentage and F_1-measure will be calculated in Excel after all the experimental runs are completed. These metrics will be based on aggregates for true positives, false positives, true negatives and false negatives summarised during the Pandas post-processing step.

The standard method to represent true positives, false positives, true negatives and false negatives in tabular form is through the use of confusion matrices. Confusion matrices typically have 2×2 dimensions with actual and predicted labels (or classes) on different axes.

These metrics are expanded upon to create McNemar contingency matrices, on which McNemar tests and Cochran's Q-tests will be based. These tests are used to compare classifier performance in ways that can be related to confidence intervals to empirically test differences between classifiers.

McNemar (1947) proposed a modified chi-squared test to identify significant differences between marginal cases in 2×2 contingency tables. In terms of classifier performance, this measurement refers to the difference between classifiers specifically in relation to incorrect classifications. The McNemar test can be used to compare two classifiers to each other, so in cases where more than two classifiers need to be compared they can either be compared pairwise in turn (that is, compare every possible combination of two classifiers from the whole set), or by comparing all the classifiers at once using Cochran's Q-test.

Cochran's Q-test (Cochran, 1950) requires a binary outcome (for classifier comparison this means that only binary classifications can be tested) and that the groups tested need to be the same size (i.e., for classifier comparison the datasets for each test run of each classifier need to be the same size).

This study makes use of the MLxtend library's (Raschka, 2018) functions to perform these tests in a Python environment. This library provides miscellaneous functions that simplify the inspection and evaluation of machine learning models through standardised means.

3.8 Limitations

This study specifically aims to compare four sentiment analysis approaches *empirically*, and does not attempt to analyse the theoretical complexity or underlying theoretical causes of the differences between these approaches. The conclusions drawn from the results may therefore be interpreted as real-world, experimental results of the use of specific implementations of these approaches in a controlled environment, instead of a generalisable result that would apply in all situations.

3.9 Summary

Having considered various research paradigms, methodologies and methods throughout this chapter, a quantitative, positivist approach was chosen based on the requirements set by the research questions and the nature of the data which was expected to be generated during experimentation. This approach would enable the use of a formal experimental methodology to compare multiple sentiment analysis techniques empirically using quantitative measurements in a controlled environment. The positivist, quantitative underpinning of the experimental design presented the only option for a study of this nature, as confirmed by the evaluation of multiple alternatives. The storage solution for this environment was also decided upon, as part of the research process, based on the needs of the study, as well as the background given in Chapter 2. The processing of the results was then discussed, followed by the limitations of this study as imposed by the choices made in this chapter.

The research environment, consisting of a combination of commondity hardware and software, in addition to purpose-built research software, will be discussed in the next chapter. This will include physical architecture and network topology of the research cluster, as well as the software running on each of the machines along with the roles they fulfil within the environment. Every component of the research software, and the part it plays in the research process will also be discussed in detail.

CHAPTER 4
RESEARCH ENVIRONMENT

4.1 Introduction

Chapter 3 discussed the research paradigm, methodology and method employed by this study, as a combination of the Positivist paradigm and the use of a formal experiment as the chosen instrument. This was followed by a discussion of the research process in the context of this study, as well as the NoSQL database back-end as a part of that process. The chapter concluded with a discussion of the analysis and interpretation of the data, and the limitations of the study as a whole.

This chapter will report on the research environment, consisting of a combination of commodity hardware and software to form a complete, standalone research cluster. This will include a discussion of the role of each machine within the cluster, and a report of the network topology between them. A discussion of both the commodity software and the research software, developed for the purposes of this study, follows. An in-depth explanation of the functions and procedures of each of the components of the research software within the data pipeline will be given.

4.2 Hardware

The hardware component of the cluster is comprised of nine Dell Optiplex 990 workstations. The specifications of the machines are given in Table 4.1.

Table 4.1: Machine Hardware Specifications and Responsibilities

Machine Number	1-8	9
Purpose	Experiment compute machine	Support services
Storage Capacity	240GB	240GB
Physical Memory	4 GB	4 GB
Processor	Intel Core i5 @ 2.4 Ghz	Intel Core i5 @ 2.4Ghz
Network Connectivity	Shared 100Mbps ethernet switch	

This means that a maximum of eight out of the nine machines could be used for the empirical experiment since one is dedicated to support services. The software components for each purpose will be discussed in the next section.

4.3 Commodity software

The software which ran on each machine differs depending on its purpose in the cluster. An overview is given in Table 4.2.

Table 4.2: Machine Software Configuration

Machine Number	1-8	9
Purpose	Experiment compute machine	Support services and network storage
Operating System	Ubuntu Server 16.04 LTS 64-bit	
Software	Apache Cassandra Puppet agent Research software NFS Client	Puppet agent NTP server DNS server NFS server

A general overview of the interaction between these machines is shown in Figure 4.1 below:

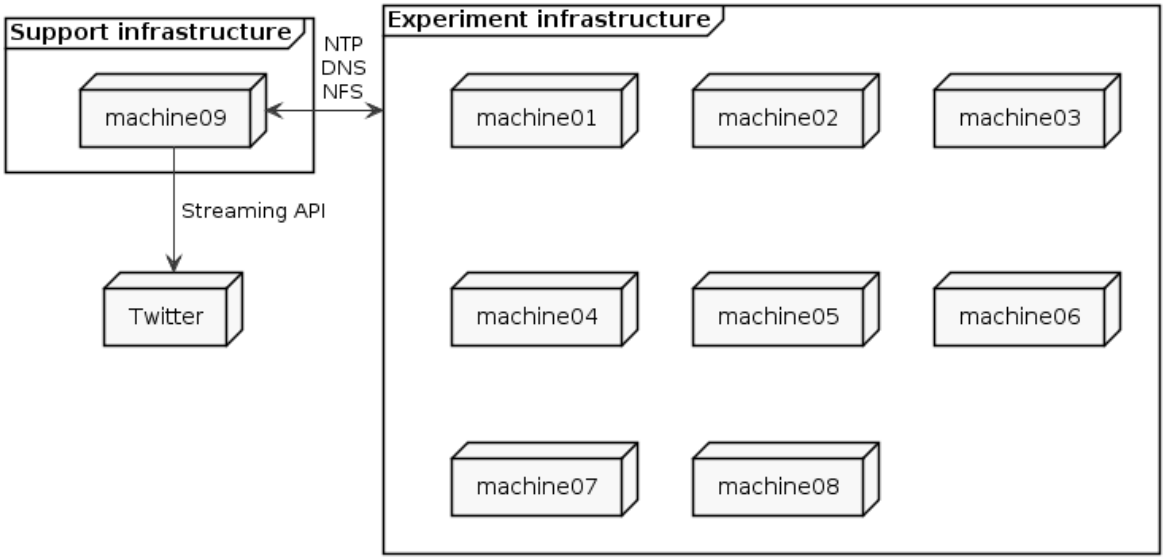

Figure 4.1: Overview of the research environment

The following sections will provide a complete discussion of the commodity software running on each of the machines. Information pertaining to the research software component can be found in section 4.3.3.

4.3.1 Machine #1-8

These machines ran the software involving the research experiment itself. In order to function correctly, each machine ran an Apache Cassandra (Apache Software Foundation, 2020) database node, an NFS client, a Puppet agent (Puppetlabs, 2020) and the research software itself.

A discussion of each software package is given below.

Puppet agent is a component that ran on every system managed by a Puppet master (which is running on an external machine for the purposes of this cluster). Puppet is a configuration management tool that enforces a specific set of configuration rules for a given collection of computers (usually referred to as a site). The Puppet agent collects the ruleset for the machine it is running on from the Puppet master and modifies the system to follow those rules. Puppet rules are organised into manifests and can include instructions to install specific software, set firewall rules and create or modify configuration files.

Apache Cassandra was discussed in the previous chapter but the implementation details on the compute machines are as follows: The Cassandra software ran on each of the eight machines. No master was defined in the configuration, since Cassandra automatically elects a master. Each node stored its own data on its local disk, and at least three copies of every table existed somewhere throughout the cluster. This gave Cassandra a higher write-throughput, since the combined write performance of multiple machines could be leveraged simultaneously.

An **NFS Client** is a network file system client that connects to the ninth (support) machine to access shared storage. This storage was used to write experiment output to a central location to simplify management.

Research Software is discussed in detail in section 4.1.3.

4.3.2 Machine #9

Machine 9 had multiple roles within the cluster. It ran the primary **NTP** and **DNS** servers for the cluster, and provided shared storage to the other machines using an NFS server.

NTP refers to the Network Time Protocol, which was used to establish the correct date and time on every machine, to ensure that their clocks are synchronised. This was crucial for a healthy Cassandra cluster, since it is very sensitive to discrepancies in timing, especially when writing data.

DNS refers to the Domain Name Service, which provided domain name resolution to all the machines in the cluster, and simplifies management by allowing distributed configurations to make use of memorable, and modifiable hostnames instead of IP addresses.

The **NFS server** was a network file server that makes a storage location on the local machine selectively available to other machines on the network. For the purposes of the experiment, the location `/var/nfs` was chosen for export. Making shared storage available on a machine that does not participate in the experiment ensured that the report writing did not incur additional I/O traffic on the local disks of the machines running the experiment, and that this machine could devote the majority of its resources to serving this purpose.

Research Software is discussed in section 4.1.3 below.

4.3.3 Research software

The research software, developed for the purposes of this study by the author, was composed of multiple parts, summarised in Table 4.3 below.

Table 4.3: Research Software

Software component	Purpose	Programming language
Tweet Collector	Opens the Twitter stream to receive and store the incoming JSON data on disk.	Go
Dataset Creator	Creates ten randomly-ordered datasets from one dataset, for the purposes of cross validation.	Python
Research Depositor	Reads the JSON Twitter data from the disk and deposits it in the Cassandra database.	Python
Research Trainer	Trains the machine learning models using data from the Cassandra database, and stores the trained models and vectorizers on disk.	Python
Research Validator	Tests each trained classifier using the validation sets created by the Dataset Creator, and records the precision, recall, f-measure and accuracy for each.	Python
Research Runner	Predicts sentiment values using the trained classifiers, compares them to the labels and outputs classified Tweets to the Cassandra database. Measures performance metrics, precision, recall, F-measure and accuracy, and outputs these values in reports.	Python

4.3.3.1 Tweet Collector

The Tweet Collector software opened a connection to the Twitter Streaming API and requests real-time streaming data that matched a predefined set of filters according to the specifications of the API (Twitter Inc., 2013). The software then stored the data received from the Twitter API on disk as raw JSON (JavaScript Object Notation) files.

This component of the research software was written in the Go programming language (Donovan & Kernighan, 2015) to take advantage of its compiled performance (as opposed to Python's interpreted performance) and native parallelisation capabilities (concepts such as channels to ferry data between running threads). A performant binary was required to ensure that the collector did not fall behind on the Twitter stream, since the API would cut off any client that does not keep up with the stream.

All of the following components were written in Python, a language popular for machine learning applications:

4.3.3.2 Dataset Creator

The Dataset Creator was responsible for creating the cross-validation files from a single set of 100 000 Tweets. These Tweets were read by the application, re-arranged in a random order and then output into new files. This process had to be completed beforehand and stored, to ensure that each technique is tested using the same set of files, and split according to the same parameters. The standard k-fold cross validator in the Scikit-learn library (Pedregosa et al., 2011) would normally generate a different random set at each run, which would have been inappropriate in this case, and would needlessly have created new datasets on the fly.

4.3.3.3 Research Depositor

The Research Depositor component read the JSON files stored by the Tweet Collector (in the case of the test set) or the Dataset Creator (in the case of the training sets), parsed the files and extracted the Tweet ID, author and text. These values were then inserted into the database at each corresponding table. For the test table, an additional row number column was added to aid with efficient distribution of the data during the run.

4.3.3.4 Research Trainer

The Researcher Trainer component loaded the 80% training data portion from each training table in turn and trained each classifier. Each classifier and its accompanying vectorizer is then stored on disk for use in the Research Validator and Research Runner, resulting in thirty classifiers and vectorizers (keeping in mind that the lexicon-based classifier is excluded from this step).

4.3.3.5 Research Validator

The research validator component loaded the 20% validation data from each training table in the database and validated each classifier created in the previous step. The resulting output, including the precision, recall, F-measure and accuracy, was then summarised in a report and written to disk.

4.3.3.6 Research Runner

The research runner loaded the 25 000 Tweets from the test table, and tested the performance of each classifier, distributed among the machines participating in the run. Each machine had an assigned machine number and was provided with the total number of machines participating

in the experiment. Using this information, combined with the values in a row number column, each machine could determine which documents to process, and skipped the rest. Each machine reported the performance metrics along with the precision, recall, F-measure and accuracy. These statistics were then combined into a single report after the run has completed.

A complete discussion of the procedures followed during the operation of the research software is given in section 5.5.

4.4 Summary

This chapter discussed the controlled research environment in which the experimental runs took place, including the complete hardware and software setup. In terms of hardware, the number of machines and the role each machine played within the cluster was reported in detail, along with the network topology used for inter-machine communication. The commodity software components were then discussed in the context of their purposes as part of the research environment. Each component of the research software written for the purposes of this study was then explored in detail, and their part in the data pipeline confirmed. Each software component (whether research or commodity) was also associated with one or more machines. This discussion covered the complete implementation of the cluster as the controlled research environment in which this study takes place.

Chapter 5 will explore the experimental design and methodology in detail. This will cover the use of empirical analysis, as opposed to theoretical analysis, as the basis of this study. The metrics commonly investigated as part of empirical analysis of algorithms are then discussed, and a subset of these metrics will then be chosen for inclusion in this study, based on the aim of the study and the limitations imposed by the equipment available. The design of the experiment as a data pipeline is then reported as a deviation from the standard machine learning process. The work distribution and data storage aspects of this pipeline is then discussed, prior to an explanation of the measurement of the proposed metrics in software. The chapter will conclude with a summary of the ethical considerations and the limitations imposed on this study by the chosen experiment design and methodology.

CHAPTER 5
EXPERIMENTAL DESIGN AND METHODOLOGY

5.1 Introduction

Chapter 3 considered various possible research methodologies and the reasoning for choosing a formal experiment. Chapter 4 discussed the research environment in terms of the various hardware and software involved to support the research experiment. These two chapters inform the experimental design and methodology

This chapter will explain the design and operation of the experiment in terms of this environment and methodology, and discuss the various statistical analyses involved in interpreting and summarising the results. Empirical analysis will be discussed as a means of evaluating the real-world performance of algorithms, as opposed to the use of theoretical analysis. The metrics commonly associated with the empirical analysis of algorithms will then be given, and a subset of these will then be chosen for inclusion in this study based on the equipment available and the aims of this study. The design of the experiment will then be reported as a data processing pipeline. This pipeline will deviate from the standard machine learning process in a number of ways, each of which will be discussed.

The work distribution and data storage aspects will then be discussed in the context of this data pipeline, followed by the methodology used to take the measurements of the metrics identified earlier. The chapter will conclude by exploring the ethical considerations involved in this study, as well as the limitations of the experiment.

5.2 Empirical analysis

As noted by a number of authors (Bartz-Beielstein, Chiarandini, Paquete, & Preuss, 2010; Papaefthymiou & Rodrigue, 1994; Sedgewick, 1998), algorithms are often compared in two ways, namely through theoretical complexity and empirical testing.

A theoretical calculation is performed based on the input size and the number of instructions executed to achieve a correct result (Cormen, Leiserson, Rivest, & Stein, 2009). This

calculation results in a function that represents the algorithmic complexity, from which it is possible to derive a theoretical asymptotic upper bound for the running time of the algorithm (Cormen et al., 2009). Afterwards an attempt is sometimes made to correlate this result with real-world performance using empirical testing.

5.3 Metrics for comparison

Empirical testing involves running the software on physical hardware and collecting performance metrics on a number of aspects (Papaefthymiou & Rodrigue, 1994). This kind of testing, when done in the field of machine learning, is often done to compare only the accuracy of the algorithms (Ghiassi et al., 2013; Khuc et al., 2012; L. Zhang et al., 2011), but it has also been used in a number of other fields to compare processing requirements (Elminaam, Abdual-Kader, & Hadhoud, 2010; Papaefthymiou & Rodrigue, 1994), memory usage (LeCun et al., 1995), training time (Aha, Kibler, & Albert, 1991; LeCun et al., 1995), network utilisation (Aha et al., 1991), throughput (how much data can be processed in a certain time frame – not to be confused with network throughput) (Elminaam et al., 2010) and power consumption (Potlapally, Ravi, Raghunathan, & Jha, 2006).

It is possible to estimate these values based on the theoretical calculations, but there is often a discrepancy between the results due to the complexity of modern computers as well as the variations between programming languages and operating systems (Bartz-Beielstein et al., 2010). A summary of these metrics is shown in Table 5.1.

Table 5.1: A summary of the performance metrics involved in empirical testing

Metric	**Purpose**	**Authors**
Accuracy	This metric influences the value associated with the results of the classifier; more accurate results allow for better informed business decisions.	(Ghiassi et al., 2013; Khuc et al., 2012; L. Zhang et al., 2011)
Processing Requirements (CPU Time)	This metric represents the processing aspect of the hardware requirements of each algorithm.	(Elminaam et al., 2010; Papaefthymiou & Rodrigue, 1994)
Memory Usage	This metric represents the volatile memory aspect of the hardware requirements of each algorithm.	(LeCun et al., 1995)
Training Time	Training time can differ between algorithms and should be accounted for during planning.	(Aha et al., 1991; LeCun et al., 1995)
Network Utilisation	Different algorithms may have a variety of bandwidth requirements that would influence hardware infrastructure planning.	(Aha et al., 1991)
Throughput	Represents the number of samples that can be classified in a certain timeframe. This should be taken into account along with the expected size of the input data stream.	(Elminaam et al., 2010)
Power Consumption	This metric has a direct effect on electricity expenses and Humidity, Ventilation and Air Conditioning (HVAC) infrastructure, since higher power consumption introduces more heat. Some companies also have targets to lower greenhouse gas emissions, which can be reduced by using less power hungry algorithms.	(Potlapally et al., 2006)

It has been noted that theoretical comparisons based on algorithmic complexity have generally been held in higher regard than empirical comparisons due to their mathematical nature which provides a measure of algorithmic performance that is not dependent on any kind of hardware or software. However, this also limits its usefulness since it cannot compensate for all the variables involved with the real-world use of these algorithms. For this reason, it is becoming more common to perform both types of analysis since it is often useful to be aware of the resource implications involved with algorithms in practice (Bartz-Beielstein et al., 2010; Papaefthymiou & Rodrigue, 1994; Sedgewick, 1998).

5.4 Chosen metrics

This study includes all of the metrics shown in Table 5.1, apart from power consumption (due to hardware limitations) and network utilisation (the algorithms tested do not make use of the network directly, only the database and shared storage have an impact on the network), to give a comprehensive view of the performance and resource requirements of all the algorithms.

5.5 Experimental design

This section will discuss the design and operation of the research software in terms of the functionality discussed in section 4.1.3 and explain the procedure followed during experimentation. This section is structured mostly according to the machine learning process as proposed by Raschka (2015), shown in Figure 5.1. The machine learning process will be discussed first, followed by how this study deviates from the standard text classification process in a number of ways.

5.5.1 Pre-processing

The following section will provide an overview of the methods available to determine the features to be used to calculate sentiment.

5.5.1.1 Feature extraction

Feature extraction, also called tokenisation, refers to the process by which features (unigrams or n-grams) are extracted from the document. A basic tokeniser can extract features on material by splitting the document by whitespace, which is used to separate words in English. Tokenizers can be extended to also consider other features such as emoticons and punctuation.

In the case of Twitter data, an additional step is required beforehand to parse the JSON data produced by the API to extract the tweet text attribute, which can then be processed by a tokenizer.

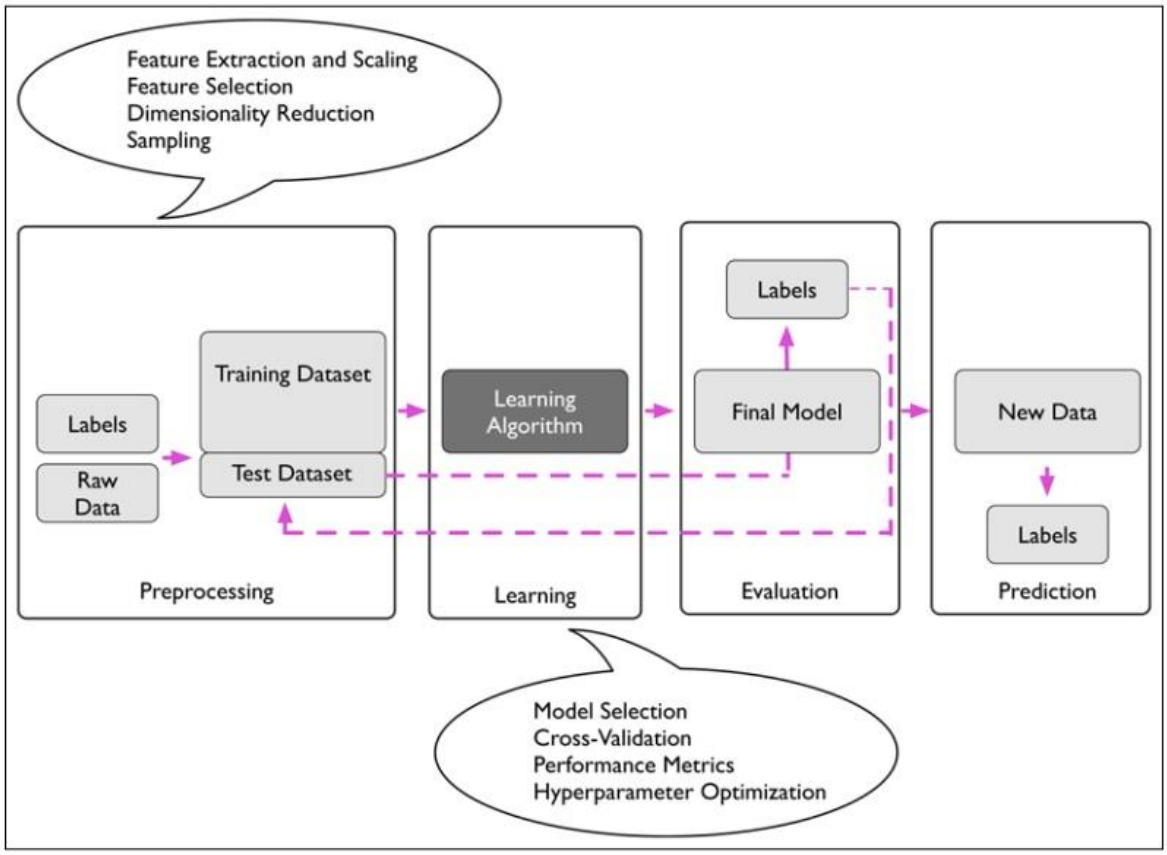

Figure 5.1: The machine learning process. Adapted from Raschka (2015)

5.5.1.2 Feature selection

Feature selection involves selecting the most important features of a text document for processing. The most prominent feature selection techniques given by Pang & Lee (2008) will be discussed in this section, followed by a motivation for the technique chosen for this project.

5.5.1.2.1 Feature vectorisation

There are generally three different ways to vectorise the features present in the training dataset:

Bag-of-words: In this scenario, only the presence of a word is recorded, in other words, the vector will have a list of binary values indicating whether a term is present or not.

Count vector: A count vectoriser records the presence of a feature along with its frequency.

Term frequency-inverse document frequency: Features may be selected based on their frequency in a document, such as is the case with term frequency-inverse document frequency

(TF-IDF), where terms are generally chosen or weighted according to their rarity. In essence, this means that words which are rare are assigned higher importance than words which are common to all documents.

5.5.1.2.2 Higher-order n-grams

Features such as bigrams or trigrams have varying results compared to unigrams. In some cases, the use of higher-order n-grams results in better classifications, while performing worse in other cases.

5.5.1.3 Dimensionality reduction

Feature selection can result in dimensionality reduction by limiting the number of features. This can be done in a number of ways, such as by selecting only the top *n* words by frequency. This reduces the size of the sparse matrix representing the words and their frequencies by document, resulting in lower training time.

5.5.1.4 Sampling

Sampling is the process whereby the raw data is split into three parts: training, validation and testing. The test dataset is removed from the raw data and set aside to be used only after the training has been completed, to ensure that the model has not been influenced by or introduced to the test data prior to the final testing.

The training and validation sets are composed of the remaining data, and usually represents 80% of the raw data combined. This 80% is usually subject to *n*-fold cross validation, whereby one dataset is repeatedly split randomly into *n* training and validation datasets. For example, 10-fold cross-validation would result in 10 different datasets, each split once again as 80% training and 20% validation, but each of the 10 datasets is simply a resampling of the original dataset training/validation dataset. The performance of the model on each sample is measured and recorded, to be summarised for a general result of the model's performance across all samples.

It is important to note that this study will deviate from the process as discussed by Raschka (2015), since the aim of this study is not to set up the ideal machine learning classifier through the use of various pre-processing and optimisation techniques, but rather to compare the

characteristics of various sentiment analysis approaches in a real-world scenario. The reasoning for, and consequences of, this is discussed in more detail in the following section.

5.5.2 Modified data pipeline

This section will reconcile the machine learning process in the previous section with the data pipeline used in this study. Figure 5.2 graphically illustrates the complete data pipeline in conjunction with the software components discussed in 4.1.3. A number of the steps in the pipeline are not directly part of the machine learning process, but remain important for this study. Each step corresponds to a specific software component, which roughly overlaps with the steps in the machine learning process mentioned previously.

Additionally, this study did not apply the hyper-parameter optimisation step. This was not performed, since the principal aim of this study is to compare the various approaches using a variety of metrics, and tuning these parameters outside of their default values may affect the other metrics in ways that make the results less generalisable. This means that no grid search was performed to find the optimal parameters for each classifier.

5.5.2.1 Step 1: Tweet Collector

Step one covers the collection of data from Twitter, through the use of the Streaming API. This was done using the Tweet Collector software, which was set to use two filters on each run, once using an English language filter and a ‘:)’ keyword filter, to collect positive English Tweets, and again using an English language filter and a ‘:(’ keyword filter, to collect negative English Tweets.

5.5.2.2 Step 2: Dataset Creator

Step two subsequently made use of the Dataset Creator program, which created the cross-validation datasets based on this input data. This step is corresponded to the pre-processing step of the machine learning process. The creation of the test dataset also happened during this step, where a remaining 25 000 Tweets were set aside and combined into a single JSON file for final testing.

5.5.2.3 Step 3: Research Depositor

At step three further pre-processing took place where the JSON data for each training, validation and test dataset was transformed into tabular data (using a crosswalk to map Twitter JSON items to database table columns) to be loaded into the database.

5.5.2.4 Step 4: Research Trainer

Step four, where the models were trained, corresponds to the learning stage of the machine learning process. This training process made use of the datasets prepared in step two. It was as part of this step that each machine learning classifier was given a matrix of values computed by a TF-IDF vectorizer function, with the 'ngram_range' parameter set to (1,2), and the 'max_features' parameter set to ten thousand (for dimensionality reduction). A value of (1,2) for the 'ngram_range' parameter means that both unigrams and bigrams were considered. The following snippet of source code briefly illustrates this process:

```
vector = TfidfVectorizer(ngram_range=(1,2),
max_features=10000)
tf_train = vector.fit_transform(x_train)
pickle.dump(vector, open('vector' + str(expNum) + '.sav',
'wb'))
```

The final line serialises the trained model and saves it to the disk for later use, allowing the same trained classifier to be re-used during the testing stage.

5.5.2.5 Step 5: Research Validator

Step five was responsible for the validation stage, where each of the ten trained models per technique was validated according to the corresponding validation dataset for that model. This is done by loading the previously saved classifier and running the 'fit()' function with the validation dataset as parameter. The best classifier of the ten trained models was then earmarked for use in the next step.

5.5.2.6 Step 6: Research Runner

Step six involved the sentiment prediction of the test dataset, which was set aside in step two for final testing. For this step, the best-performing classifier from each technique was used to predict sentiment values for the same set of 25 000 Tweets. Each classifier was tested ten times to determine whether the output of the classifier was deterministic (*i.e.* the input directly

determines the output without any element of randomness), and to establish the average performance metrics across multiple runs. This involved loading the already-trained classifier from the disk and using it to predict the sentiment values of the test dataset. The test dataset was loaded from the Cassandra database table by each machine, where each one used their assigned machine number, and the total number of machines partaking in the experiment, to determine which samples to classify, so as to ensure that the workload is distributed as equally as possible, and to ensure that no sample is classified twice.

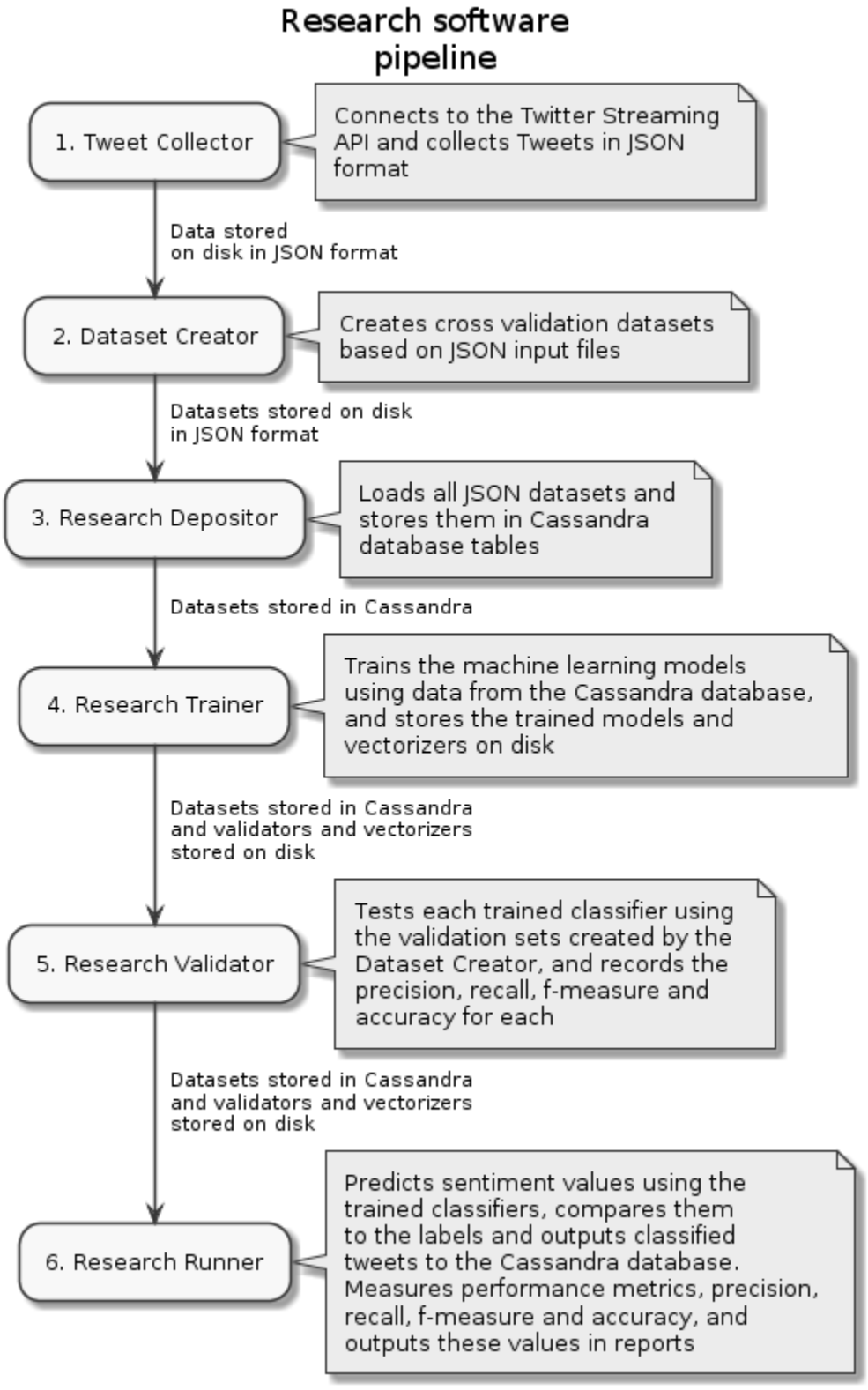

Figure 5.2: The research software pipeline

5.6 Work distribution

At the test stage the classification work was distributed between the machines participating in the experiment at any given time. The number of machines could range between three and eight, which means that the data distribution can be documented as shown in Table 5.2:

Table 5.2: Work distribution

Machine count	Tweets per machine	Caches per machine	Caches per core	Estimated final cache size (Tweets)
3	8333.3	17.0	4.3	333.3
4	6250.0	13.0	3.3	250.0
5	5000.0	10.0	2.5	500.0
6	4166.7	9.0	2.3	166.7
7	3571.4	8.0	2.0	71.4
8	3125.0	7.0	1.8	125.0

Each machine had four CPU cores, and each cache contained 500 Tweets. The caches per machine value is the mathematical ceiling of the number of Tweets per machine divided by the cache size, this ensures that any remainder is included in a cache, even if the cache is not full. Figure 5.3 illustrates the deviation of the number of Tweets from the nearest 500.

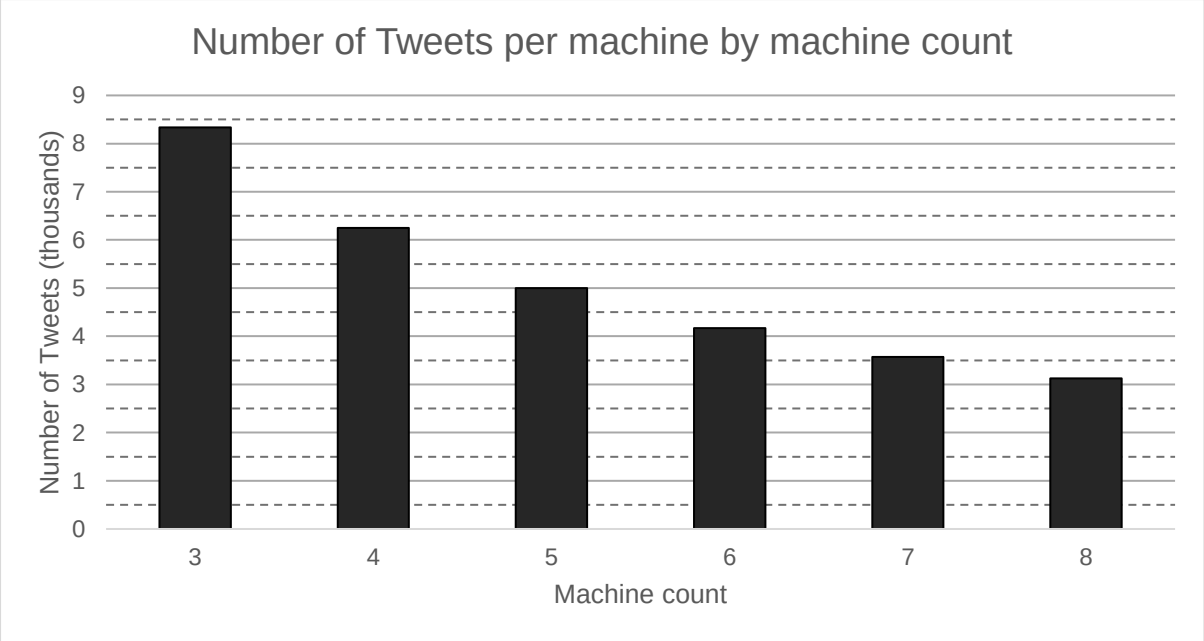

Figure 5.3: Number of Tweets per machine by machine count

Depending on the efficiency of the classifier used, an increase in per-machine performance may be visible when the final cache on each machine is closer to full, since each cache requires the creation of a new thread, reducing the overall efficiency per cache. The final cache size (Tweets) for each machine at every cluster size is given in the last column. Based on this distribution, it was expected that the highest efficiency will be found when using cluster sizes consisting of three or five machines.

5.7 Data storage

The data is stored in multiple Cassandra database tables consisting of the following fields:

- **UUID**: A unique identifier assigned by the database engine to serve as a primary key.
- **Row number**: A row number which uniquely identifies the tweet, generated during the data deposit stage, and running consecutively from one up to the total number of Tweets in the table. This field is exclusive to the test table (as opposed to the training tables) to allow the machines to divide the work equally based on the row number and the machine number.
- **Tweet ID**: This is the unique 64-bit integer identifier which is assigned by Twitter when the Tweet is posted.
- **Tweet text**: This field contains the raw text as it was posted to Twitter.
- **Tweet author**: This field contains the screenname for the person who posted the tweet.
- **Date & Time**: This field contains the date and time the tweet was posted.
- **Sentiment classification**: This field contains the sentiment of the tweet determined by the sentiment analysis technique. It is represented as a Boolean field indicating whether the tweet is positive. That is, a positive tweet will have the value ‘true’ and a negative tweet will have the value ‘false’. A different field type could be chosen if more than two classifications are required.

These fields were represented with a corresponding data type in the database, as shown in Table 5.3.

Table 5.3: Database table fields

Field name	Cassandra field type
UUID	timeuuid
Row number	bigint
Tweet ID	bigint
Tweet text	text
Tweet author	text
Date and time	timestamp
Sentiment classification	boolean

This storage was generally meant to serve as an example which may be used in a real-world scenario, where the results of the classification may need to be stored for future use. The tables created for the purpose of serving the raw data and storing the output are listed in Table 5.4.

Table 5.4: Database tables

Table name	Purpose	Includes row number field
Output	Stores the Tweets classified in each experiment	No
Test	Stores the 25 000 Tweets used for testing	Yes
Train0	Stores the first set of training data	No
Train1	Stores the second set of training data	No
Train2	Stores the third set of training data	No
Train3	Stores the fourth set of training data	No
Train4	Stores the fifth set of training data	No
Train5	Stores the sixth set of training data	No
Train6	Stores the seventh set of training data	No
Train7	Stores the eighth set of training data	No
Train8	Stores the ninth set of training data	No
Train9	Stores the tenth set of training data	No

The complete Cassandra Query Language (CQL) commands used to create the tables containing the training data, test data and output are listed in Appendix A.

5.8 Measurements

Subsets of the metrics discussed in chapter 4 were measured for each technique at the training and test stages. This section will discuss the metrics that applied to each stage, and how they were measured.

5.8.1 Training measurements

Training applies to Naïve-Bayes, Support Vector Machine and Neural Network classifiers. The AFINN lexicon-based classifier is excluded from these measurements since its use does not require the user to train a machine learning model.

Training was only performed on one machine, as opposed to the evaluation which made use of a distributed environment. This is due to the fact that the Scikit-learn Python library (Pedregosa et al., 2011) only supports single-machine training by default. The library also does not report fine-grained metrics for resource usage during the training phase. As a result, all measurements for training were taken using the GNU time utility (version 1.7).

This tool was applied throughout the training process by prefixing each training run with the following command:

```
\time -v
```

The backslash prevents the bash built-in function '`time`' from overriding GNU time, which resides at `/usr/bin/time`. This was done because the bash built-in function does not have the capability to record and report the required metrics. The `-v` flag provides the additional reporting for the following metrics (copied here verbatim from the command output), with measurements of interest to this study marked in bold:

- Command being timed
- **User time (seconds)**
- **System time (seconds)**
- **Percent of CPU this job got**
- **Elapsed (wall clock) time (h:mm:ss or m:ss)**
- Average shared text size (kbytes)
- Average unshared data size (kbytes)
- Average stack size (kbytes)
- Average total size (kbytes)
- **Maximum resident set size (kbytes)**
- Average resident set size (kbytes)

- Major (requiring I/O) page faults
- Minor (reclaiming a frame) page faults
- Voluntary context switches
- Involuntary context switches
- Swaps
- File system inputs
- File system outputs
- Socket messages sent
- Socket messages received
- Signals delivered
- Page size (bytes)
- Exit status

User time represents the time the process spent in user mode (a non-privileged mode), also called ‘user space’. System time is the number of seconds the process spent executing system calls inside the kernel (privileged mode). Elapsed time is the actual time the command took to execute from start to finish.

Maximum resident size refers to the amount of memory the process occupies in the main physical memory, as opposed to the virtual size which would include other factors such as swap space.

5.8.2 Validation measurements

Once each classifier has been trained, it is validated using the remaining 20% of the training dataset. This procedure is followed to determine the general performance of the classifier in multiple scenarios using cross validation. For these purposes, only the following metrics are included:

- Precision
- Recall
- F-measure
- Accuracy as percentage

These statistics are determined using the `accuracy_score` and `classification_report` functions of the Scikit-learn `metrics` package. In the case of Precision, Recall and F-measure, the values are the weighted averages for the two classes (positive and negative). See Section 3.3.2.4 for more details on these metrics.

5.8.3 Test measurements

Unlike the training phase, the test phase is more easily controlled from the application's side. Instead of making use of the built-in methods for determining the accuracy of the classifier, the application will run a prediction on each input, and measure the metrics discussed in Chapter 4 accordingly. This also solves the problem where the AFINN classifier's interface is incompatible with that of the Scikit-learn classifiers.

5.9 Ethical considerations

This study makes use of actual data posted online by individual Twitter users. The data collected is processed in aggregate and no attempts are made to identify any specific individuals. The use of this data is in line with the terms and conditions, and privacy policy of Twitter Inc (Twitter, 2020b). which permits the use of any data published publicly on the platform for research purposes. Any person using the platform has given their explicit agreement of the terms and conditions and privacy policy during registration and through their continued use of the service. This study has also been reviewed and approved by the Ethics Committee of the Faculty of Natural and Agricultural Sciences and a copy of the letter of approval is available on the CD accompanying this study and is also included in this text as Appendix B.

5.10 Limitations

The methodology proposed in this study is well-suited to a practical, empirical comparison between approaches, and will allow for an evaluation of the various approaches in an environment that mimics a real-world application. No attempts will be made to investigate the theoretical complexity or underlying theoretical causes of the empirical results, since such examinations are outside the scope of this study.

5.11 Summary

This chapter described the experimental design and methodology followed in the study, including the background of empirical analysis and theoretical analysis, and the reasoning behind the choice to make use of empirical analysis for this study. The metrics commonly used to evaluate algorithms empirically were discussed, and a subset of these were then chosen for inclusion in this study, based on the equipment available and the aims of this study. The technical details surrounding the techniques used to measure the required metrics at the training, validation and testing stages of the process were then reported. The complete data pipeline to be used during the experiment was discussed in detail, in addition to the standard machine learning workflow and how this study deviated from it. This included the pre-processing steps required to transform the data received from Twitter into usable data files, creating reusable datasets for n-fold cross validation, depositing this data into the distributed database and finally proceeding with the training, validation and testing. The work distribution at every cluster size was documented and discussed, along with the technical details of the data storage used. The ethical considerations pertaining to a study of this nature was also reported, followed by a discussion of the limitations imposed upon the study by the choice of experimental design and methodology.

The next chapter will document and discuss the results obtained using this procedure. These results will be given as a breakdown by approach and metric, prior to an overall comparison of the metrics between these approaches during training and testing. The chapter will conclude with the testing of the hypotheses stated earlier.

CHAPTER 6
RESULTS

6.1 Introduction

Chapter 5 provided an explanation of the experimental design and statistical methods involved with the processing of the data generated during the experiment. This included the use of empirical analysis, as opposed to theoretical analysis, for the purposes of this study. This was followed by a discussion of the metrics commonly used to evaluate algorithms empirically, and the metrics chosen for inclusion in this study, based on the aims of the study and the equipment available. The experimental design was then reported as a data pipeline which deviated from the standard machine learning pipeline, with reasons given for this deviation. From there, the work distribution and data storage aspects of the pipeline were discussed, followed by the techniques used to measure the metrics proposed earlier, during each phase of the study. The chapter concluded by discussing the ethical considerations involved in a study making use of social media data and the limitations imposed by the experimental design and methodology.

This chapter will provide a summary of the results obtained along with a detailed explanation thereof, divided according to the techniques used. The data will be reported as a breakdown by approach and metric, prior to being combined for overall analysis at the training and testing levels. In all cases the metrics reported are also given at each cluster size, from three to eight machines. The performance of the database is also documented and discussed. This will be followed by testing of the hypotheses stated earlier in the dissertation. The raw input and output data from these experiments are given as supplementary material and is not included in this document.

6.2 Sentiment analysis approaches

Four popular sentiment analysis approaches were tested as part of this study. This included a lexicon-based technique (which does not make use of machine learning), a multinomial Naïve-Bayes classifier, a Support Vector Machine classifier and a Neural Network classifier. The metrics obtained by training the models and performing predictions using these techniques are documented in this section.

6.2.1 Lexicon-based

The lexicon-based classifier made use of the AFINN Python library discussed in Chapter 3. This classifier was only tested, not trained or validated, since it is not a classifier that makes use of machine learning, but instead relies on a lexicon. This section documents the results of the testing and discusses these results in detail.

6.2.1.1 Testing metrics

The AFINN classifier was applied to the 25 000-tweet test set, for ten runs at each of the cluster sizes between three and eight machines. Note that these runs are not validation runs, but rather test runs making use of the exact same dataset. Validation (and therefore cross-validation) did not take place due to the fact that this is not a machine learning classifier.

Table 6.1: Lexicon-based classifier performance metrics for three machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,274548	0,000022	0,001545	818,1
2	0,272653	0,000020	0,001415	818,2
3	0,268982	0,000021	0,001302	818,1
4	0,273436	0,000023	0,001486	818,0
5	0,274989	0,000022	0,001446	818,1
6	0,272748	0,000021	0,001519	818,0
7	0,270544	0,000021	0,001443	818,1
8	0,276006	0,000022	0,001436	818,2
9	0,273642	0,000022	0,001343	818,1
10	0,273960	0,000021	0,001448	818,0
Average	**0,273151**	**0,000021**	**0,001438**	**818,1**

With the use of three machines, as shown in Table 6.1, the classifier displays very consistent results, with minor variations on database time being the most noticeable.

Table 6.2: Lexicon-based classifier performance metrics for four machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,274445	0,000022	0,001612	818,5
2	0,277696	0,000021	0,001575	818,5
3	0,275496	0,000024	0,001851	818,5
4	0,271210	0,000023	0,001518	818,5
5	0,275598	0,000021	0,001683	818,5
6	0,273066	0,000021	0,001717	818,5
7	0,274306	0,000022	0,001562	818,5
8	0,272545	0,000022	0,001562	818,5
9	0,272193	0,000022	0,001589	818,5
10	0,274026	0,000021	0,001563	818,5
Average	**0,274058**	**0,000022**	**0,001623**	**818,5**

This consistency shown with three machines is continued when a fourth machine is added, as shown in Table 6.2, albeit with some increase in CPU time and maximum resident size.

Table 6.3: Lexicon-based classifier performance metrics for five machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,282822	0,000021	0,001759	819,5
2	0,280262	0,000020	0,001630	819,5
3	0,280706	0,000020	0,001585	819,5
4	0,280078	0,000022	0,001556	819,5
5	0,283882	0,000021	0,001670	819,5
6	0,281055	0,000022	0,001571	819,5
7	0,284420	0,000024	0,001552	819,5
8	0,283928	0,000022	0,001493	819,5
9	0,284607	0,000023	0,001677	819,5
10	0,283170	0,000022	0,001546	819,5
Average	**0,282493**	**0,000022**	**0,001604**	**819,5**

All metrics remain consistent across the board as the fifth machine is added, as shown in Table 6.3, with the database time increasing slightly compared to the four machine results.

Table 6.4: Lexicon-based classifier performance metrics for six machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,286431	0,000025	0,001846	819,1
2	0,280450	0,000024	0,001765	819,1
3	0,277953	0,000023	0,001716	819,2
4	0,282309	0,000024	0,001617	819,2
5	0,279201	0,000023	0,001710	819,1
6	0,279674	0,000022	0,001701	819,1
7	0,281793	0,000022	0,001712	819,1
8	0,279426	0,000022	0,001651	819,1
9	0,276786	0,000021	0,001678	819,2
10	0,277836	0,000022	0,001620	819,2
Average	**0,280186**	**0,000023**	**0,001702**	**819,1**

The difference with the addition of a sixth machine is negligible, as shown in Table 6.4, as the results remain very consistent across all ten runs.

Table 6.5: Lexicon-based classifier performance metrics for seven machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,284446	0,000021	0,001755	819,6
2	0,284720	0,000021	0,001686	819,6
3	0,283970	0,000021	0,001580	819,6
4	0,284154	0,000021	0,001581	819,6
5	0,285667	0,000021	0,001985	819,6
6	0,283914	0,000020	0,001715	819,6
7	0,283202	0,000022	0,001692	819,6
8	0,279559	0,000020	0,001647	819,6
9	0,284843	0,000021	0,001587	819,6
10	0,284243	0,000021	0,001597	819,6
Average	**0,283872**	**0,000021**	**0,001682**	**819,6**

The above-mentioned consistency persists as a seventh machine is added, as shown in Table 6.5.

Table 6.6: Lexicon-based classifier performance metrics for eight machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,280984	0,000020	0,001702	819,3
2	0,283525	0,000021	0,001636	819,3
3	0,282543	0,000021	0,001712	819,3
4	0,281452	0,000022	0,001632	819,3
5	0,280498	0,000019	0,001757	819,3
6	0,283123	0,000021	0,001671	819,3
7	0,282577	0,000022	0,001575	819,3
8	0,283232	0,000021	0,001691	819,3
9	0,282138	0,000021	0,001552	819,4
10	0,283640	0,000022	0,001599	819,3
Average	**0,282371**	**0,000021**	**0,001653**	**819,3**

And, finally, the addition of the eighth machine cements the consistency of the classifier across all four metrics, with very minor changes between all the machine counts, as shown in Table 6.6. A summary of the average values for each metric measured at each machine count is shown in Table 6.7 below:

Table 6.7: Average lexicon-based performance metrics for all machine counts

Machine count	CPU time	Duration	Database time	Max RS (MB)
3	0,273151	0,000021	0,001438	818,1
4	0,274058	0,000022	0,001623	818,5
5	0,282493	0,000022	0,001604	819,5
6	0,280186	0,000023	0,001702	819,1
7	0,283872	0,000021	0,001682	819,6
8	0,282371	0,000021	0,001653	819,3

These numbers can be graphically represented as a chart for each metric:

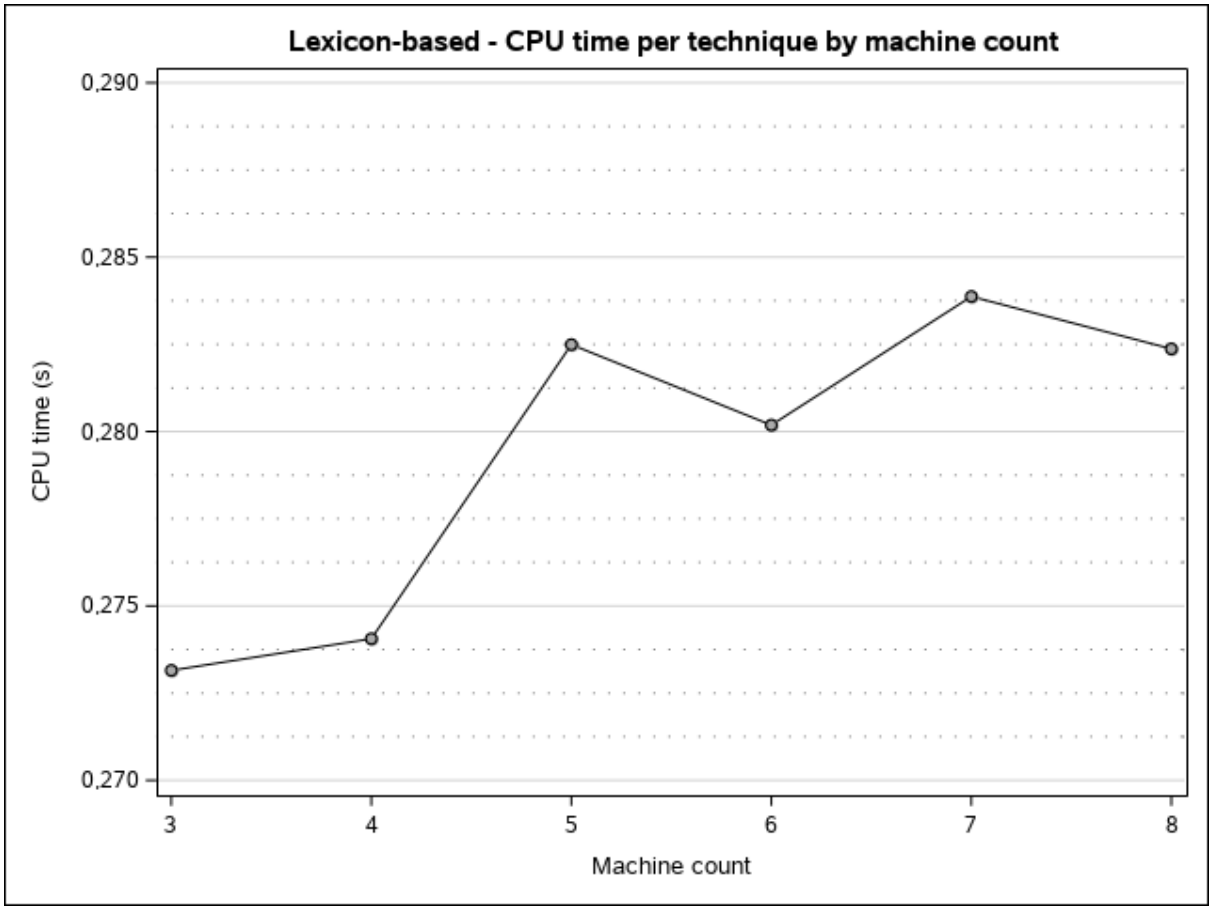

Figure 6.1: Average lexicon-based CPU time by machine count

The lexicon-based classifier displays a minor increase in CPU time with the participation of five machines, from where it gradually reduces prior to increasing again from six machines onwards, as shown in Figure 6.1.

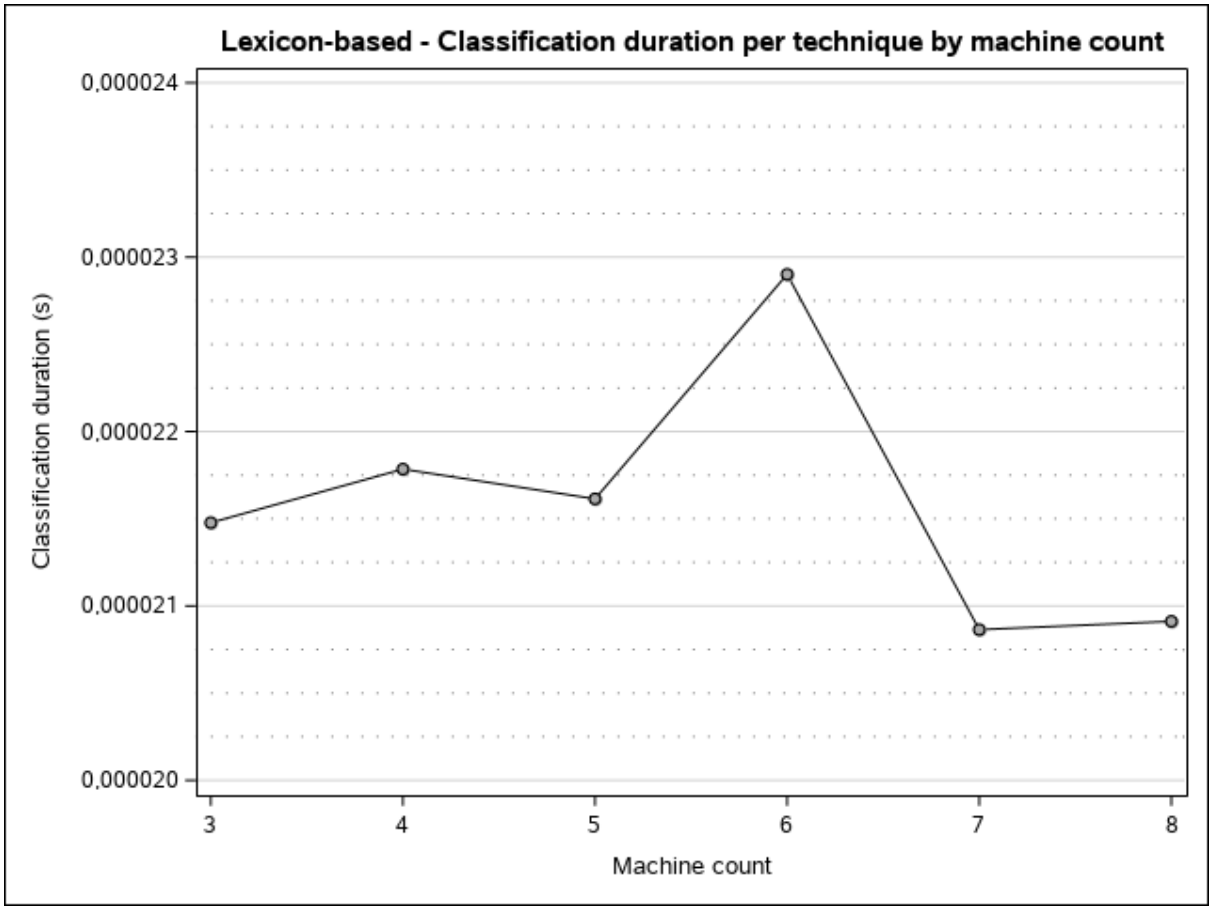

Figure 6.2: Average lexicon-based duration by machine count

Figure 6.2 shows a minor increase in classification duration with the participation of between four and six machines, before levelling off for the remainder of the experiments. The duration for each classification is miniscule, and therefore the time resolution to measure differences is quite low, leading to a graph that exaggerates the minor differences between the values.

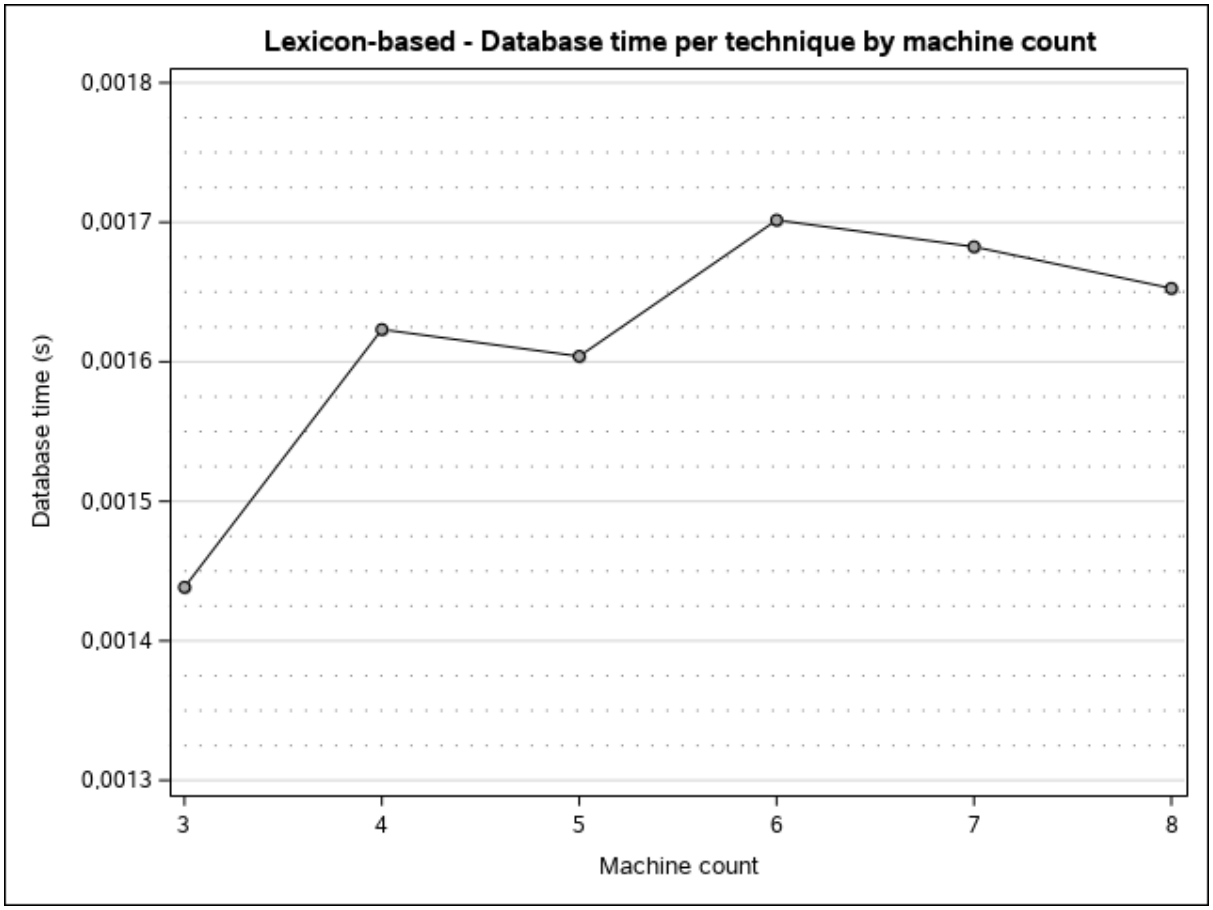

Figure 6.3: Average lexicon-based database time by machine count

The database time for the lexicon-based classifier is shown in Figure 6.3. An increase in time is shown up to six machines, from where the measurement appears to nearly level out up to eight machines.

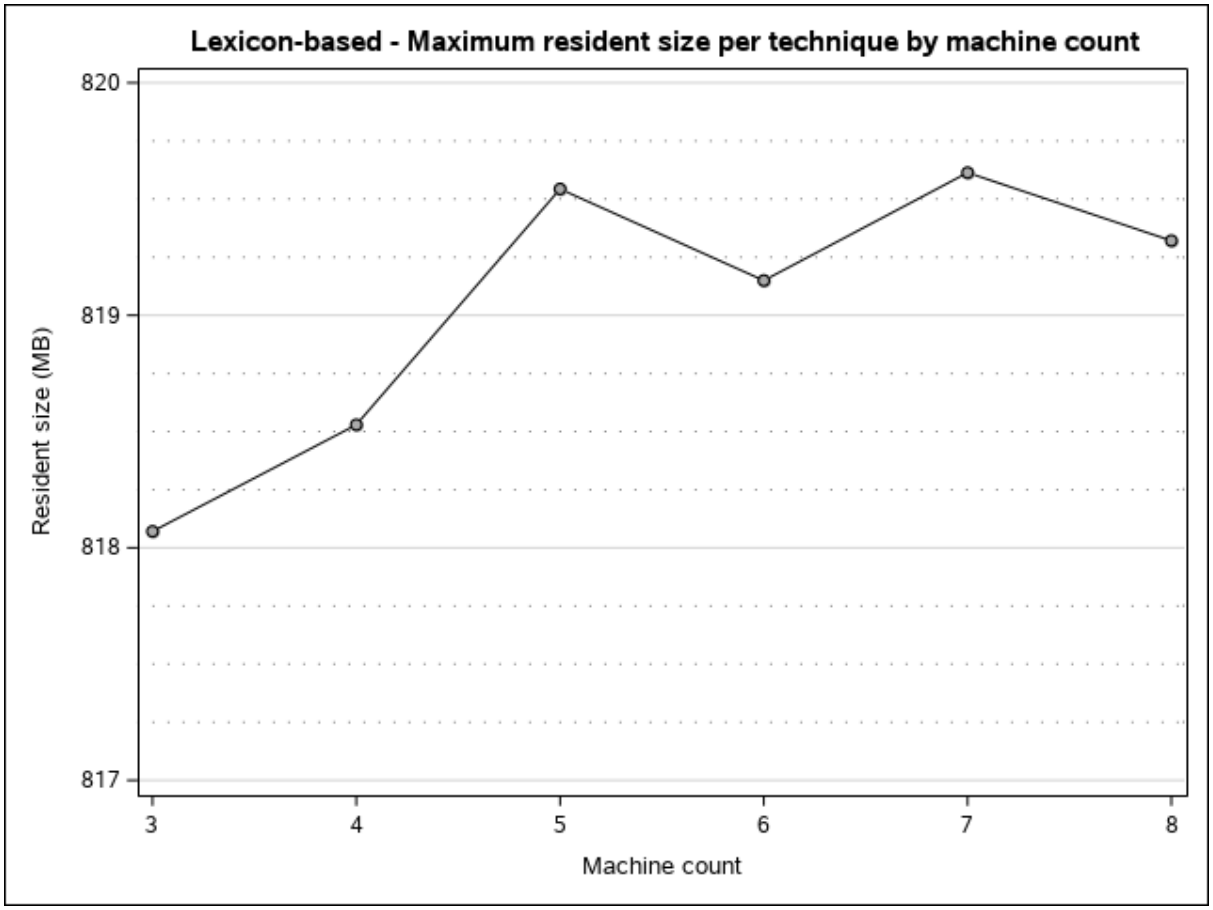

Figure 6.4: Average lexicon-based maximum resident size by machine count

The classifier displays no clear pattern as the number of machines are increased, as shown in Figure 6.4. From five machines onwards, the maximum resident size appears to level off.

Summarised classifier performance metrics for the ten runs are given in Table 6.8 below:

Table 6.8: Lexicon-based classifier performance metrics

Run #	False Positives	False Negatives	True Positives	True Negatives
1	3914	4234	7300	9552
2	3914	4234	7300	9552
3	3914	4234	7300	9552
4	3914	4234	7300	9552
5	3914	4234	7300	9552
6	3914	4234	7300	9552
7	3914	4234	7300	9552
8	3914	4234	7300	9552
9	3914	4234	7300	9552
10	3914	4234	7300	9552

These results indicate that the trained classifier's classification is deterministic (given the same input, it will produce the same output), since the results are identical across all ten runs and all six machine counts. The computed values for precision, recall, F-measure and accuracy are given in Table 6.9.

Table 6.9: Summary of Lexicon-based classifier performance metrics

Precision	Recall	F-Measure	Accuracy
0.650972	0.69287683	0.67127106	0.67408

A high recall indicates that the classifier performs better when identifying true negatives, in contrast to a poorer precision metric, which reflects the reduced performance concerning the identification of true positives. These values combined lead to an F-measure of roughly 0.67, and an overall accuracy of about 67%.

6.2.2 Naïve-Bayes

The Naïve-Bayes classifier was trained, validated and then tested using the Twitter data collected previously. The measurements taken during these steps are discussed in the following sections.

6.2.2.1 Training metrics

The Naïve-Bayes classifier completed ten runs (cross validation runs), training on 80 000 Tweets. The results are shown in Table 6.10.

Table 6.10: Naïve-Bayes training metrics

Experiment #	Usr time (s)	Sys (*) time (s)	CPU%	Elapsed (seconds)	Max RS (**) (KB)
1	8,03	0,46	71,00	11,80	322608,00
2	8,10	0,60	73,00	11,90	322520,00
3	8,20	0,48	72,00	12,00	321360,00
4	8,00	0,52	71,00	11,80	322632,00
5	8,35	0,55	74,00	12,00	320920,00
6	7,50	0,57	70,00	11,40	320952,00
7	8,01	0,54	73,00	11,70	321988,00
8	7,94	0,55	72,00	11,70	321752,00
9	8,10	0,65	72,00	12,10	321708,00
10	8,16	0,54	74,00	11,70	321368,00
Average	**8,04**	**0,56**	**72,33**	**11,81**	**321688,89**
Standard deviation	**0,22**	**0,05**	**1,32**	**0,20**	**648,04**

(*) Sys corresponds to System time.
(**) Max RS refers to maximum resident size.

The experiment numbers for all techniques correspond to the table numbers in Cassandra, offset by one. For example, the data for experiment one is stored in table `Twitter.train0`. Each classifier and vectorizer are also available in the supplementary material, as `*.sav` files. CPU time corresponds to the user time mentioned in section 5.8.

Training the Multinomial Naïve-Bayes classifier takes on average 12 seconds, of which most of the time (roughly 93%) is spent in unprivileged mode. This indicates that nearly ten per cent of time is devoted to system calls - which can generally not be optimised out of the algorithm. Training on 80 000 Tweets in such a short period of time is a favourable result, since it means that much larger datasets could likely be trained in a reasonable timeframe. The standard deviation is small, indicating relatively consistent performance for the input datasets.

The CPU usage is under 100%, which suggests that the algorithm is I/O bounded in this case. At just over 70% CPU usage per run, this suggests that a further 25% to 30% performance improvement could be gained by simply increasing the I/O bandwidth. Under a constant, uninterrupted stream of Tweets it is expected that the CPU usage would be closer to 100%. This also indicates that the Scikit-learn training algorithm is single-threaded, since CPU usage under Linux is measured as multiples of 100% according to the number of threads (or cores) available. A system with four cores/threads would therefore only be saturated at a reported 400%.

The maximum resident size is also fairly constant at roughly 320 Megabytes for each run. This is a promising result since such a low memory requirement makes the technique more accessible to users with less powerful computing hardware.

6.2.2.2 Evaluation metrics

The remaining 20 000 Tweets each training table were then used to validate each corresponding classifier. The results of these ten validation runs are shown in Table 6.11.

Table 6.11: Naïve-Bayes evaluation metrics

Experiment #	Precision	Recall	F-measure	Accuracy %
1	0,88	0,87	0,87	86,86%
2	0,88	0,87	0,87	87,11%
3	0,87	0,87	0,86	86,57%
4	0,87	0,87	0,86	86,55%
5	0,88	0,87	0,87	87,01%
6	0,88	0,87	0,87	87,23%
7	0,88	0,87	0,87	87,17%
8	0,88	0,87	0,87	86,70%
9	0,88	0,87	0,87	86,90%
10	0,88	0,87	0,87	86,99%
Average	**0,88**	**0,87**	**0,87**	**86,91%**
Standard deviation	**0,00**	**0,00**	**0,00**	**0,24%**

The evaluation metrics show similar consistency to the training metrics. Precision and recall was nearly equivalent, indicating that the technique is consistent in its performance identifying true positives and true negatives. The resulting F-measure reflects this consistency as well. The accuracy was generally consistent at about 86-87%, with a minor standard deviation. The sixth run attained the highest accuracy, at 87.23%.

6.2.2.3 Testing metrics

The best-performing classifier (number six) was the used to perform the final tests. It was applied to the test set containing 25 000 Tweets (20% of the total of 125 000 Tweets) for ten consecutive runs on eight machines.

Summarised performance values for these ten runs are given in the following tables:

Table 6.12: Naïve-Bayes performance metrics for three machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,336860	0,000026	0,001888	863,3
2	0,334511	0,000023	0,001650	863,3
3	0,331185	0,000022	0,001808	863,3
4	0,330867	0,000023	0,001574	863,3
5	0,337138	0,000022	0,001779	863,3
6	0,332967	0,000024	0,001612	863,3
7	0,328222	0,000022	0,001829	863,3
8	0,330048	0,000022	0,001757	863,3
9	0,334795	0,000023	0,001561	863,3
10	0,332173	0,000023	0,001566	863,3
Average	**0,332877**	**0,000023**	**0,001702**	**863,3**

Performance across the ten runs is generally consistent, with very little variation in any of the metrics. The amount of time spent in user space is low, which suggests that the execution was I/O bound. The duration is consistent, as is the database time. In this case the database is time two orders of magnitude longer than the classification time, most likely in part due to the network latency involved with the distributed database transaction. The maximum resident size is very consistent, as expected given that the same input is used for each run.

Table 6.13: Naïve-Bayes performance metrics for four machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,334514	0,000024	0,001833	863,9
2	0,333736	0,000025	0,001707	863,9
3	0,331784	0,000024	0,001553	863,9
4	0,339712	0,000026	0,001798	863,9
5	0,339963	0,000026	0,001666	863,9
6	0,334557	0,000026	0,001725	863,9
7	0,340288	0,000027	0,001677	863,9
8	0,337918	0,000026	0,001580	863,9
9	0,338283	0,000026	0,001675	863,9
10	0,335654	0,000025	0,001677	863,9
Average	**0,336641**	**0,000025**	**0,001689**	**863,9**

The difference between three and four machines in terms of measurements is miniscule, and this difference only increases negligibly in the case of the database time as the number of

machines is increased to eight. The following four tables document the metrics for the remaining machine counts, followed by a summary and discussion.

Table 6.14: Naïve-Bayes performance metrics for five machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,347086	0,000026	0,001765	864,8
2	0,347526	0,000025	0,001641	864,8
3	0,343765	0,000025	0,001548	864,8
4	0,342384	0,000025	0,001570	864,8
5	0,346593	0,000027	0,001712	864,8
6	0,340106	0,000024	0,001618	864,8
7	0,346907	0,000026	0,001613	864,8
8	0,344102	0,000026	0,001580	864,8
9	0,346932	0,000026	0,001654	864,8
10	0,344409	0,000025	0,001750	864,8
Average	**0,344981**	**0,000026**	**0,001645**	**864,8**

All figures remain consistent between runs when five machines take part in the experiment. Fluctuations compared to four machines remain negligible.

Table 6.15: Naïve-Bayes performance metrics for six machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,343770	0,000023	0,002895	864,2
2	0,337918	0,000023	0,002167	864,2
3	0,340075	0,000025	0,001886	864,2
4	0,338942	0,000025	0,001901	864,2
5	0,340671	0,000025	0,001874	864,2
6	0,337798	0,000026	0,001854	864,2
7	0,335892	0,000026	0,001902	864,2
8	0,338797	0,000027	0,001799	864,2
9	0,339651	0,000026	0,001796	864,2
10	0,334704	0,000025	0,001708	864,2
Average	**0,338822**	**0,000025**	**0,001978**	**864,2**

The addition of a sixth machine does not have a major impact on any of the metrics compared to a cluster size of five. Performance and resource usage remain consistent between runs.

Table 6.16: Naïve-Bayes performance metrics for seven machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,347849	0,000026	0,002106	864,9
2	0,347965	0,000026	0,001965	864,9
3	0,345848	0,000026	0,001713	864,9
4	0,344715	0,000025	0,001766	864,9
5	0,343859	0,000025	0,001710	864,9
6	0,343782	0,000025	0,001707	864,9
7	0,349754	0,000025	0,001727	864,9
8	0,344460	0,000025	0,001594	864,9
9	0,346027	0,000025	0,001836	864,9
10	0,344785	0,000025	0,001704	864,9
Average	**0,345904**	**0,000025**	**0,001783**	**864,9**

Performance and resource usage with the addition of a seventh machine remain consistent across the board. No major differences are apparent compared to a six machine cluster.

Table 6.17: Naïve-Bayes performance metrics for eight machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,340897	0,000022	0,002820	864,6
2	0,339915	0,000024	0,002087	864,7
3	0,337253	0,000022	0,001851	864,6
4	0,332762	0,000022	0,001825	864,6
5	0,340571	0,000023	0,001685	864,6
6	0,338428	0,000023	0,001679	864,6
7	0,341325	0,000024	0,001596	864,7
8	0,343945	0,000025	0,001528	864,7
9	0,337964	0,000024	0,001598	864,6
10	0,339552	0,000023	0,001520	864,6
Average	**0,339261**	**0,000023**	**0,001819**	**864,6**

With the addition of the eighth machine, the results remain very consistent across the board, with only very minor variation between runs, and even between the additions of new machines to the experiment. A summary of the average values for each metric measured at each machine count is shown in Table 6.18 below:

Table 6.18: Average Naïve-Bayes performance metrics for all machine counts

Machine count	CPU time	Duration	Database time	Max RS (MB)
3	0,332877	0,000023	0,001702	863,3
4	0,336641	0,000025	0,001689	863,9
5	0,344981	0,000026	0,001645	864,8
6	0,338822	0,000025	0,001978	864,2
7	0,345904	0,000025	0,001783	864,9
8	0,339261	0,000023	0,001819	864,6

These figures can be graphically represented as a chart for each metric:

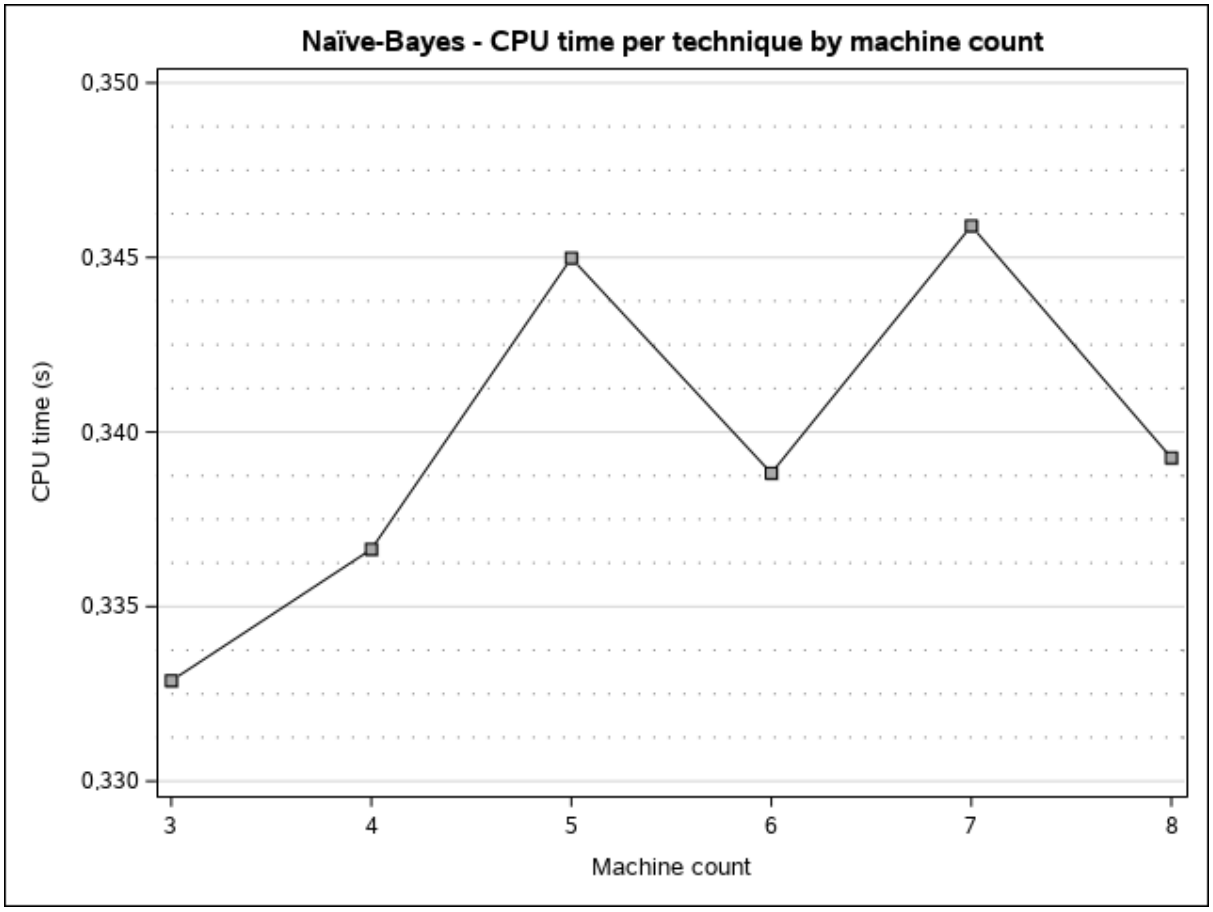

Figure 6.5: Average Naïve-Bayes CPU time by machine count

No clear relationship between CPU time and machine count emerges from Figure 6.5, with very little variation between them. This suggests that there is a very low impact on CPU time as the number of machines increase. The measurements represent small time windows, causing minor deviations to appear amplified.

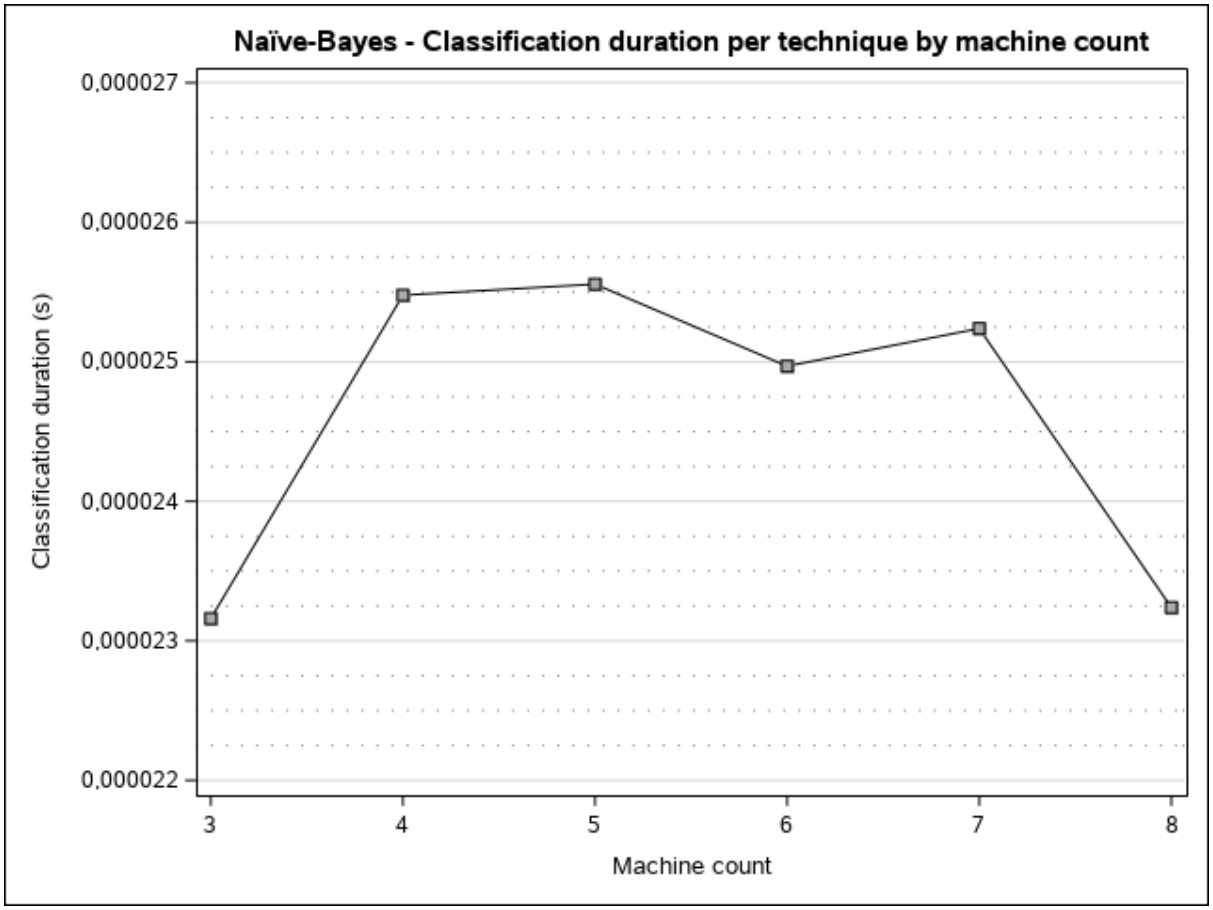

Figure 6.6: Average Naïve-Bayes duration by machine count

The same applies to the relationship between average duration and machine count, where Figure 6.6 displays a very minor deviation in duration across the range of machines. The actual value measured is miniscule, and accuracy is likely limited by the mechanism used to measure the timespan for each tweet classification. It appears that the time duration spent for classification is unaffected by the number of machines involved in the experiment. This can be expected, as there is no connection between the classification steps of the machines, although they could be affected by the load caused by the combined traffic on the distributed database. In this case, that effect appears to be so small as to be undetectable.

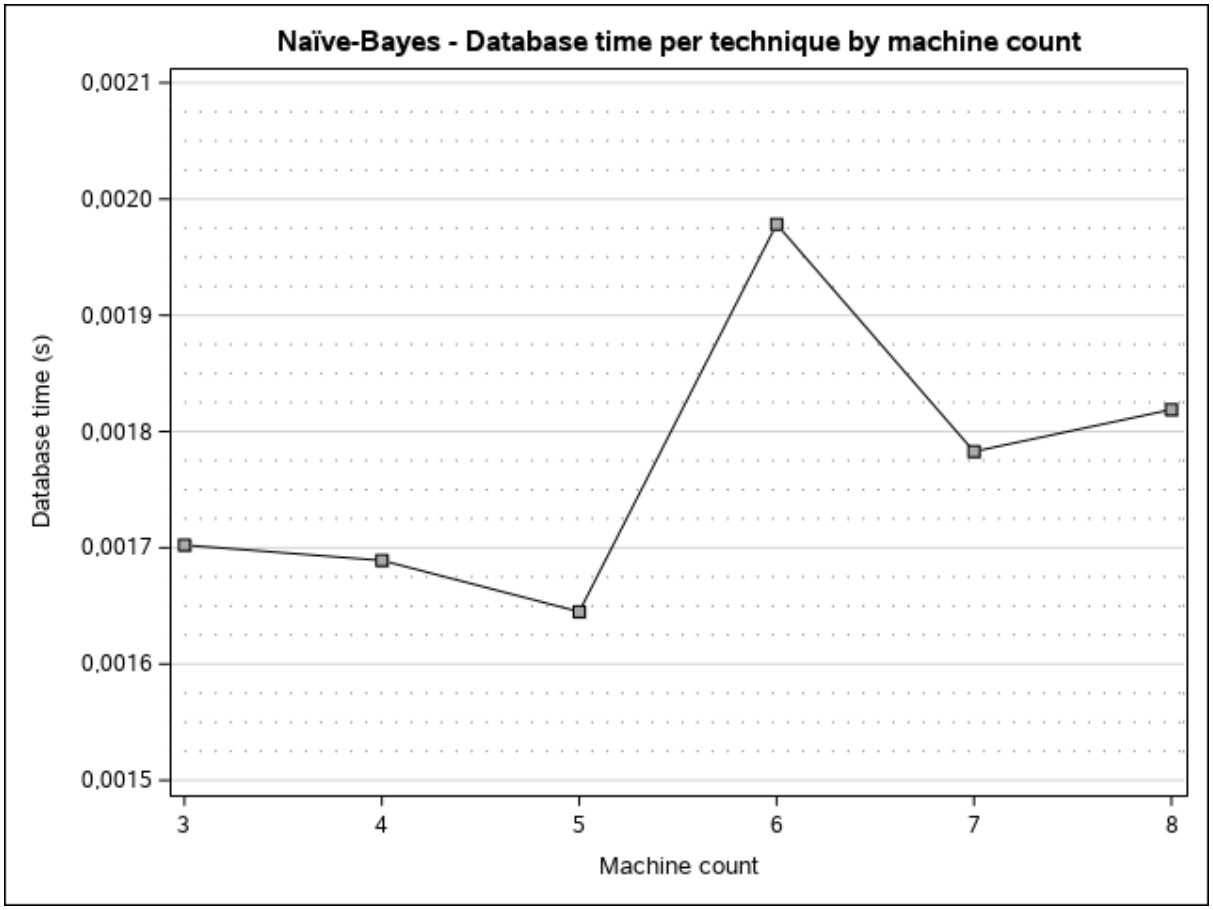

Figure 6.7: Average Naïve-Bayes database time by machine count

Figure 6.7 shows a variable database time, with no clear relationship with the number of machines. Section 6.4.2 discusses database performance across all techniques and machine counts in more detail.

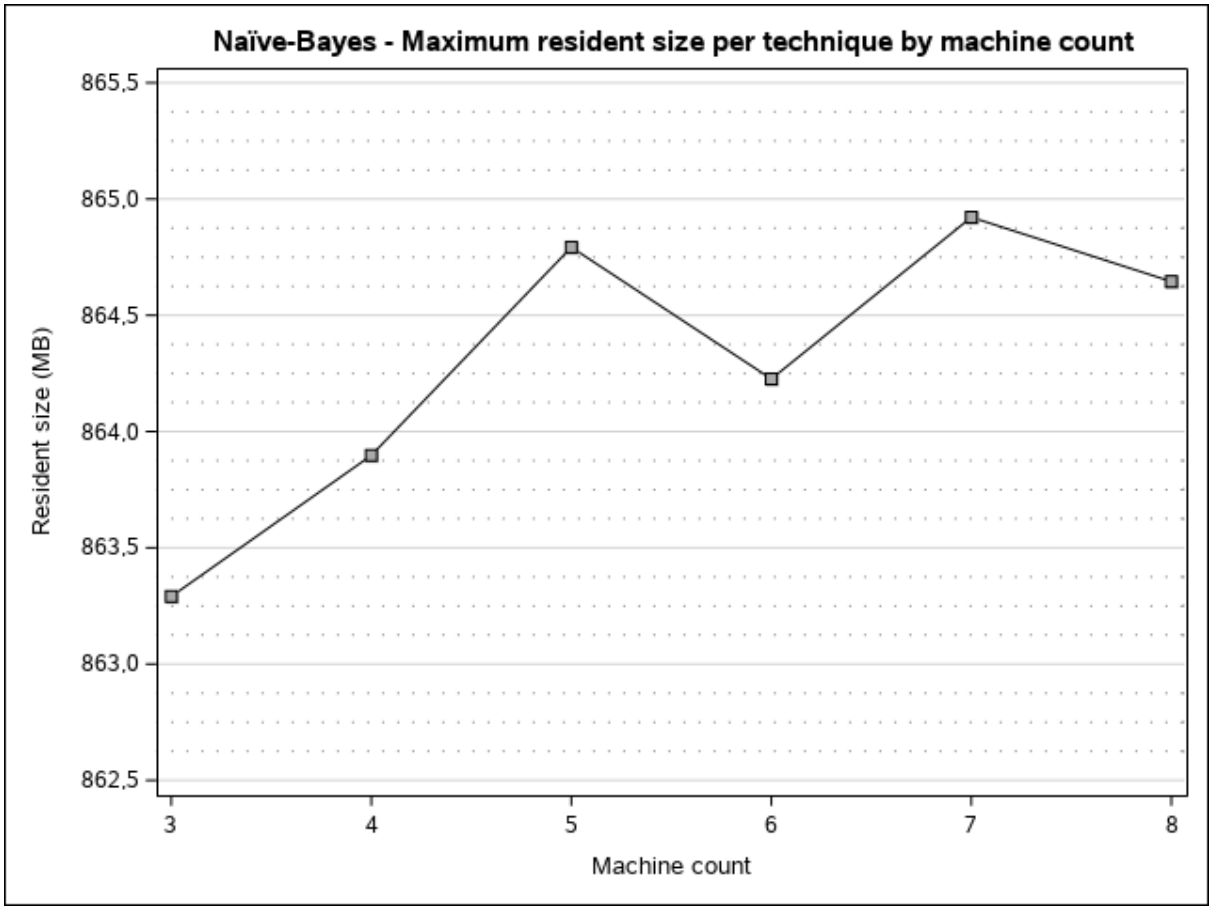

Figure 6.8: Average Naïve-Bayes maximum resident size by machine count

A general increase in maximum resident size is visible in Figure 6.8, with some variation visible at the upper range. Overall, Naïve-Bayes appears to scale very well as the number of machines increase, especially at the higher end above four machines.

Summarised classifier performance metrics for the ten runs are given in Table 6.19 below.

Table 6.19: Naïve-Bayes classifier performance metrics

Run #	False Positives	False Negatives	True Positives	True Negatives
1	683	3439	8095	12783
2	683	3439	8095	12783
3	683	3439	8095	12783
4	683	3439	8095	12783
5	683	3439	8095	12783
6	683	3439	8095	12783
7	683	3439	8095	12783
8	683	3439	8095	12783
9	683	3439	8095	12783
10	683	3439	8095	12783

These results indicate that the trained classifier's classification is deterministic (given the same input, it will produce the same output), since the results are identical across all ten runs and all six machine counts. The computed values for precision, recall, F-measure and accuracy are given in Table 6.20.

Table 6.20: Summary of Naïve-Bayes classifier performance metrics

Precision	Recall	F-Measure	Accuracy
0.922192	0.788004	0.849833	0.83512

A high precision indicates that the classifier performs better when identifying true positives, in contrast to a poorer recall metric, which reflects the reduced performance concerning the identification of true negatives. These values combined lead to an F-measure of approximately 85%, and an overall accuracy of 83.5%.

6.2.3 Support Vector Machine

This section will discuss the results obtained during training, validation and testing of the Support Vector Machine classifier, in terms of performance and accuracy.

6.2.3.1 Training metrics

The Support Vector Machine classifier completed ten runs (cross validation runs), training on 80 000 Tweets. The results are shown in Table 6.21.

Table 6.21: Support Vector Machine training metrics

Experiment #	Usr time (s)	Sys time (s)	CPU%	Elapsed (seconds)	Max RS (KB)
1	911,58	1,24	99,00	906,86	524140,00
2	934,53	1,16	99,00	940,34	530432,00
3	943,33	1,08	99,00	949,22	528588,00
4	921,28	1,18	99,00	926,54	525000,00
5	912,70	1,20	99,00	917,56	521964,00
6	922,14	0,99	99,00	927,16	530376,00
7	919,24	1,12	99,00	924,24	525280,00
8	922,36	1,22	99,00	927,55	524404,00
9	908,67	1,12	99,00	913,90	525756,00
10	916,54	1,21	99,00	921,67	529656,00
Average	**921,24**	**1,15**	**99,00**	**925,50**	**526559,60**
Standard deviation	**10,63**	**0,08**	**0,00**	**12,26**	**2973,94**

6.2.3.2 Evaluation metrics

Each trained classifier was then cross-validated using 20 000 Tweets remaining from each corresponding split. The results of this validation are shown in Table 6.22.

Table 6.22: Support Vector Machine evaluation metrics

Experiment #	Precision	Recall	F-measure	Accuracy %
1	0,87	0,84	0,84	84,42%
2	0,87	0,85	0,84	84,68%
3	0,86	0,84	0,83	83,74%
4	0,86	0,84	0,83	83,53%
5	0,86	0,84	0,84	84,35%
6	0,86	0,84	0,84	84,41%
7	0,86	0,84	0,84	84,35%
8	0,86	0,84	0,84	84,03%
9	0,86	0,84	0,84	84,19%
10	0,86	0,84	0,83	83,89%
Average	**0,86**	**0,84**	**0,84**	**84,16%**
Standard deviation	**0,00**	**0,00**	**0,00**	**0,36%**

The results are consistent across the ten runs, with minor deviations accuracy. The second run achieved the maximum accuracy by a slight margin, reported at 84.68%.

6.2.3.3 Testing metrics

Classifier number two achieved the highest accuracy among the ten classifiers trained. It was then applied to the test set containing 25 000 Tweets (20% of the total of 125 000 Tweets) for ten consecutive runs on three to eight machines.

Table 6.23: Support Vector Machine performance metrics for three machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	3,534658	0,000037	0,002964	884,6
2	3,547478	0,000037	0,002831	884,6
3	3,541904	0,000037	0,002890	884,6
4	3,544508	0,000037	0,002897	884,6
5	3,536751	0,000037	0,002887	884,6
6	3,546736	0,000036	0,002804	884,6
7	3,543037	0,000039	0,002903	884,6
8	3,553971	0,000037	0,002831	884,6
9	3,544459	0,000037	0,002855	884,6
10	3,560355	0,000037	0,002779	884,6
Average	**3,545386**	**0,000037**	**0,002864**	**884,6**

The classifier displays low overall variability at a cluster size of three machines. Classification duration is highly consistent.

Table 6.24: Support Vector Machine performance metrics for four machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	3,519055	0,000038	0,002935	885,2
2	3,516823	0,000040	0,002990	885,2
3	3,508245	0,000037	0,003109	885,2
4	3,520846	0,000037	0,002887	885,2
5	3,518490	0,000037	0,002957	885,2
6	3,529183	0,000038	0,002948	885,2
7	3,524615	0,000040	0,002978	885,2
8	3,532358	0,000037	0,002900	885,2
9	3,514136	0,000039	0,002984	885,2
10	3,523882	0,000039	0,002933	885,2
Average	**3,520763**	**0,000038**	**0,002962**	**885,2**

The consistency shown at three machines carries over to four machines, with a minor increase in database time.

Table 6.25: Support Vector Machine performance metrics for five machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	3,530346	0,000039	0,002929	886,1
2	3,535971	0,000037	0,002786	886,1
3	3,532940	0,000038	0,002727	886,1
4	3,535248	0,000037	0,002757	886,1
5	3,529964	0,000035	0,002844	886,1
6	3,537796	0,000038	0,002741	886,1
7	3,526917	0,000038	0,002747	886,1
8	3,523895	0,000037	0,002835	886,1
9	3,520578	0,000037	0,002759	886,1
10	3,531628	0,000037	0,002751	886,1
Average	**3,530528**	**0,000038**	**0,002787**	**886,1**

The addition of a fifth machine to the cluster incurs a slight increase in resident size, but remains consistent across all ten runs.

Table 6.26: Support Vector Machine performance metrics for six machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	3,480901	0,000041	0,003212	885,6
2	3,476149	0,000040	0,003626	885,6
3	3,482687	0,000040	0,003290	885,6
4	3,494196	0,000040	0,003179	885,6
5	3,496256	0,000041	0,003160	885,6
6	3,491191	0,000042	0,003272	885,6
7	3,482916	0,000041	0,003371	885,6
8	3,484729	0,000043	0,003221	885,6
9	3,479995	0,000040	0,003412	885,6
10	3,489855	0,000041	0,003262	885,6
Average	**3,485887**	**0,000041**	**0,003300**	**885,6**

A cluster size of six displays the same consistency across the ten runs, with a minor reduction in resident size, along with a small reduction in CPU time.

Table 6.27: Support Vector Machine performance metrics for seven machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	3,515314	0,000041	0,002953	886,3
2	3,514863	0,000041	0,002837	886,3
3	3,528361	0,000041	0,002724	886,2
4	3,525126	0,000040	0,002819	886,2
5	3,525790	0,000041	0,002838	886,2
6	3,518485	0,000042	0,002893	886,3
7	3,526658	0,000042	0,002758	886,2
8	3,507568	0,000042	0,002868	886,2
9	3,528023	0,000040	0,002940	886,2
10	3,514148	0,000039	0,002839	886,2
Average	**3,520434**	**0,000041**	**0,002847**	**886,2**

The addition of a seventh machine continues the trend of consistent results across all ten runs, with very minor variations compared to six machines.

Table 6.28: Support Vector Machine performance metrics for eight machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	3,521243	0,000041	0,003501	886,0
2	3,543571	0,000040	0,002788	886,0
3	3,544687	0,000041	0,002720	886,0
4	3,549084	0,000040	0,002733	886,0
5	3,555336	0,000041	0,002682	886,0
6	3,539774	0,000038	0,003006	886,0
7	3,547852	0,000039	0,002852	886,0
8	3,517909	0,000039	0,002989	886,0
9	3,519705	0,000040	0,002828	886,0
10	3,563114	0,000039	0,002743	886,0
Average	**3,540227**	**0,000040**	**0,002884**	**886,0**

The use of eight machines displays no major departure from the results obtained in the previous fifty runs, with very consistent results across all ten runs. A summary of the average values for each metric measured at each machine count is shown in Table 6.29 below:

Table 6.29: Average Support Vector Machine performance metrics for all machine counts

Machine count	CPU time	Duration	Database time	Max RS (KB)
3	3,545386	0,000037	0,002864	884,6
4	3,520763	0,000038	0,002962	885,2
5	3,530528	0,000038	0,002787	886,1
6	3,485887	0,000041	0,003300	885,6
7	3,520434	0,000041	0,002847	886,2
8	3,540227	0,000040	0,002884	886,0

These figures can be graphically represented as a chart for each metric:

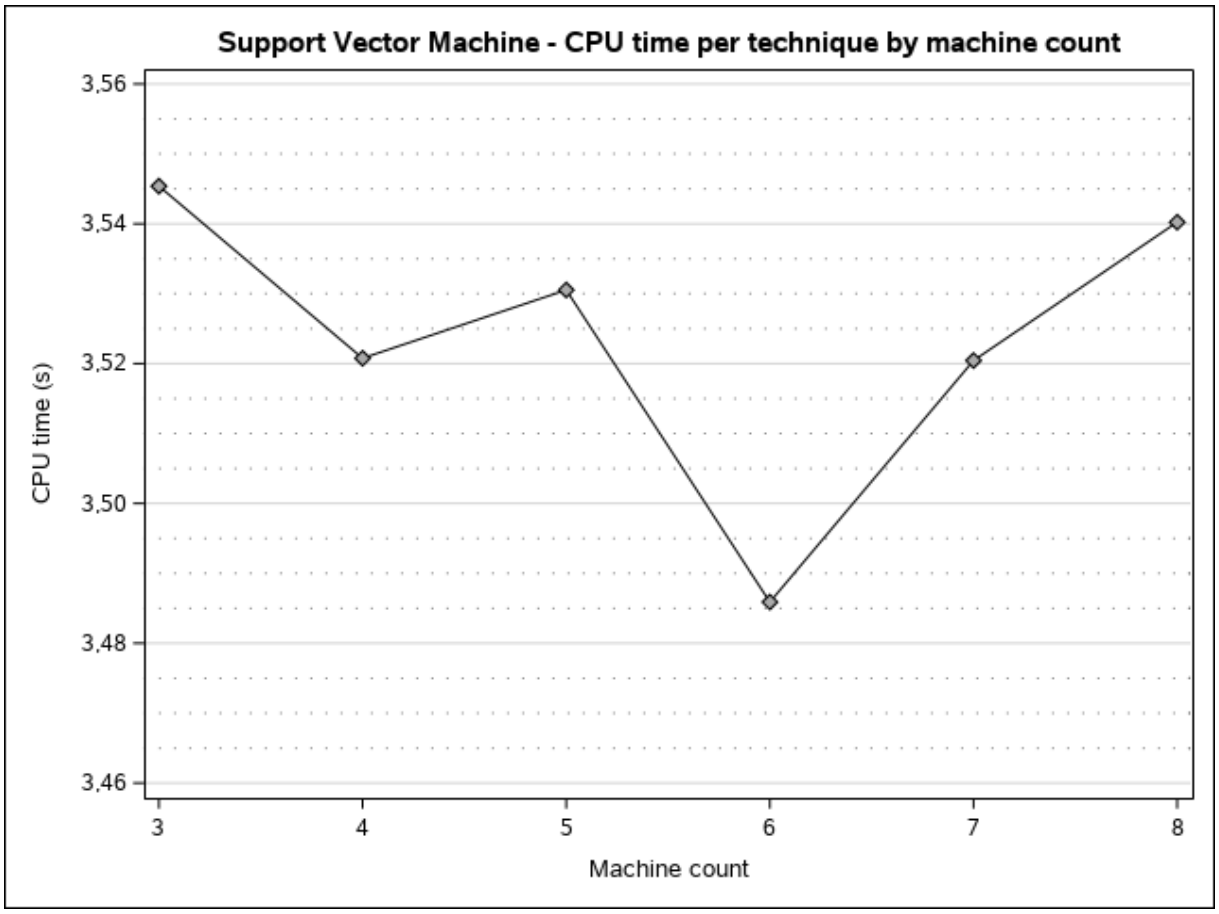

Figure 6.9: Average SVM CPU time by machine count

The SVM classifier is considerably more CPU-intensive compared to the Naïve-Bayes classifier during classification. It also displays a higher variation in CPU-time, as shown in Figure 6.9.

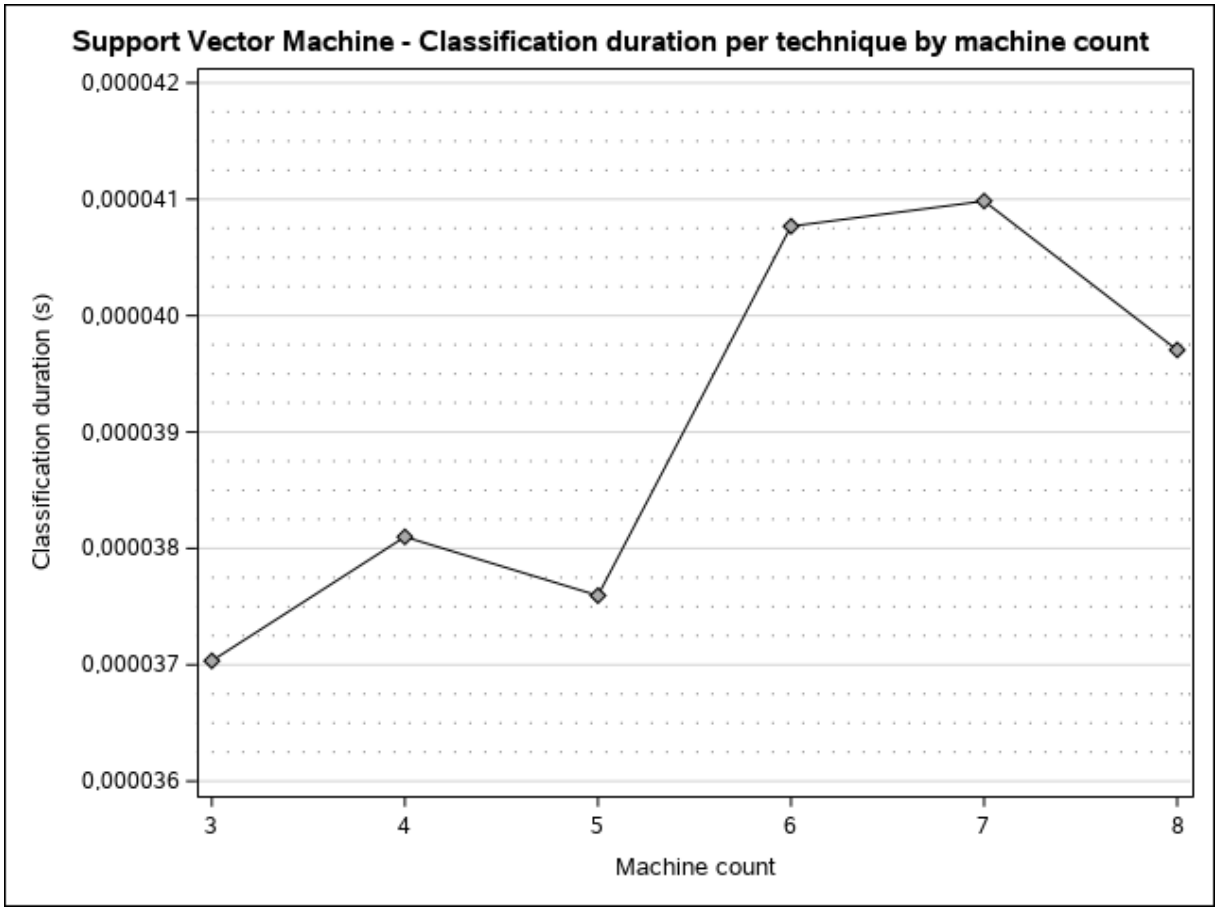

Figure 6.10: Average SVM duration by machine count

Classification duration is generally stable as the number of machines are increased, as shown in Figure 6.10. A gradual increase in duration is apparent, with a slight reduction at the introduction of the eighth machine.

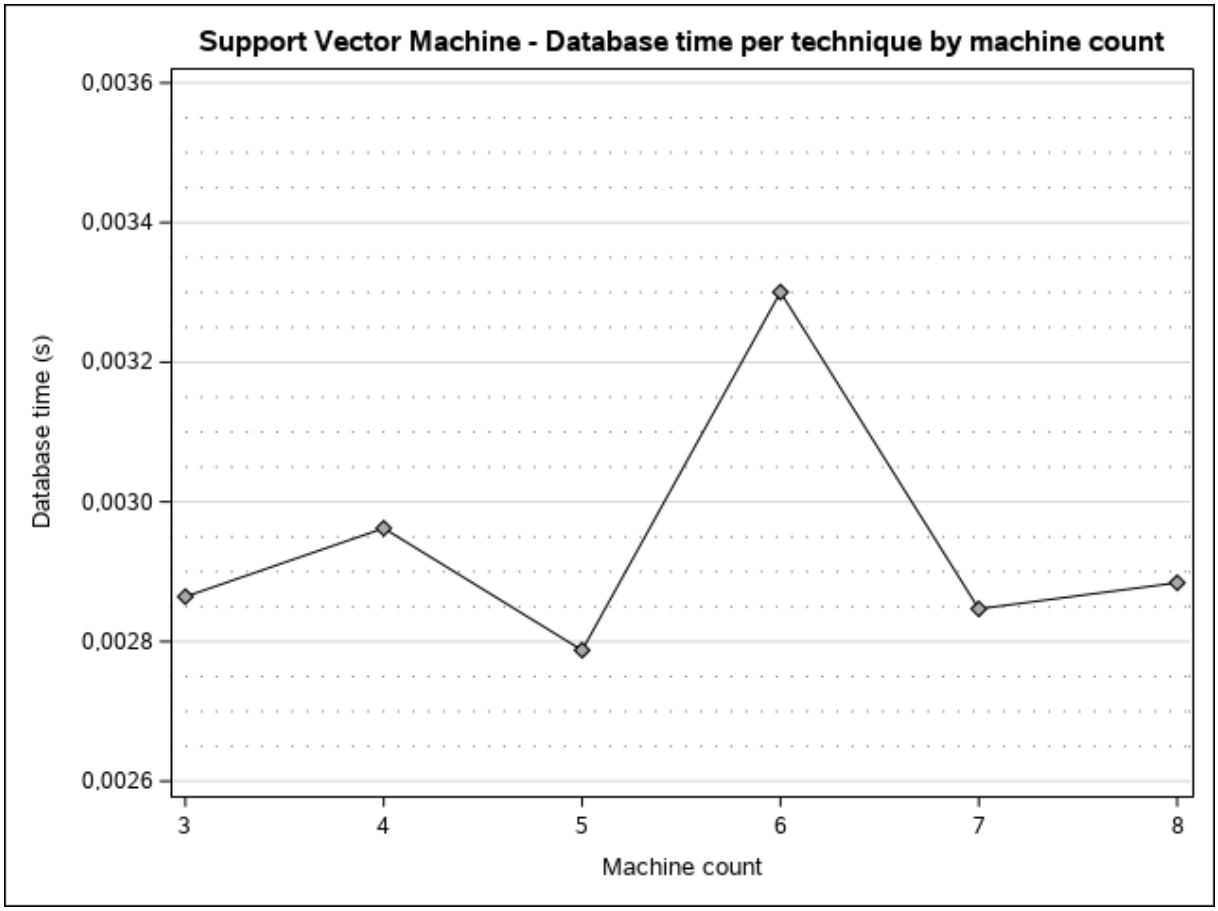

Figure 6.11: Average SVM database time by machine count

The database time shown in Figure 6.11 exhibits a similar pattern to the one displayed by the Naïve-Bayes classifier in Figure 6.7. No clear pattern emerges from the figure, and the variation in the values are negligibly small.

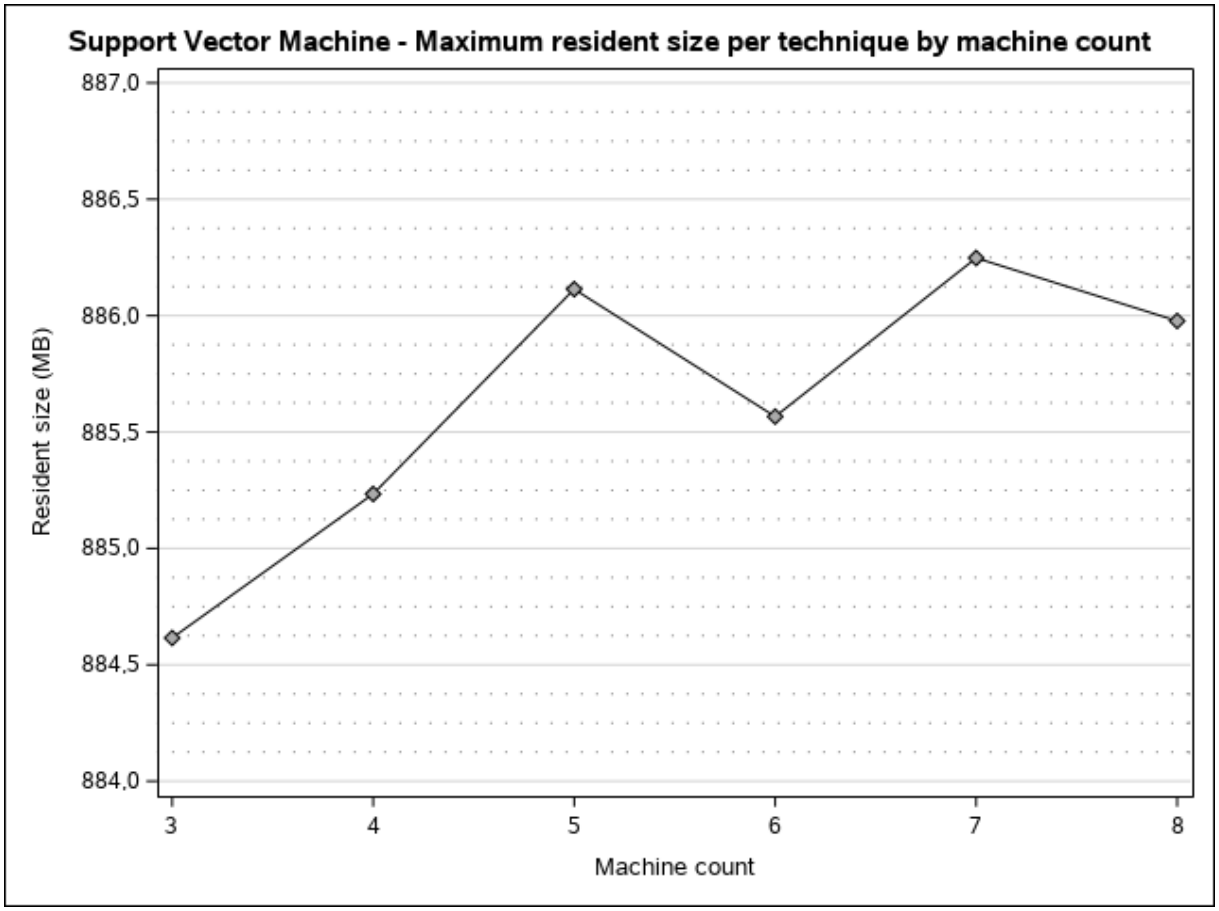

Figure 6.12: Average SVM maximum resident size by machine count

The average maximum resident size occupied by the SVM classifier is shown in Figure 6.12. There is a general increase in relation with the number of machines participating, although it appears to level off past five machines. With a range of only two megabytes between runs, the variation is minor compared to the total usage.

Summarised classifier performance metrics for the ten runs are given in Table 6.30 below.

Table 6.30: Support Vector Machine classifier performance metrics

Run #	False Positives	False Negatives	True Positives	True Negatives
1	413	4950	6584	13053
2	413	4950	6584	13053
3	413	4950	6584	13053
4	413	4950	6584	13053
5	413	4950	6584	13053
6	413	4950	6584	13053
7	413	4950	6584	13053
8	413	4950	6584	13053
9	413	4950	6584	13053
10	413	4950	6584	13053

These results indicate that the trained classifier's classification is deterministic (given the same input, it will produce the same output), since the results are identical across all ten runs and all six machine counts. The calculated values for precision, recall, F-measure and accuracy are given in Table 6.31.

Table 6.31: Summary of Support Vector Machine classifier performance metrics

Precision	Recall	F-Measure	Accuracy
0.9409747	0.7250458	0.8190173	0.78548

A high precision indicates that the classifier performs better when identifying true positives, in contrast to a poorer recall metric, which reflects the reduced performance concerning the identification of true negatives. These values combined lead to an F-measure of approximately 82%, and an overall accuracy of 78.5%.

6.2.4 Neural Network

This section will discuss the results obtained during training, validation and testing of the Neural Network classifier, in terms of performance and accuracy.

6.2.4.1 Training metrics

The Neural Network classifier completed ten runs (cross validation runs), training on 80 000 Tweets. The results are shown in Table 6.32.

Table 6.32: Neural Network training metrics

Experiment #	Usr time (s)	Sys time (s)	CPU%	Elapsed (seconds)	Max RS (KB)
1	2363,50	1,96	99,00	2370,46	376180,00
2	3579,56	2,49	99,00	3584,88	400764,00
3	1882,67	1,86	99,00	1889,27	369012,00
4	3439,76	2,32	99,00	3444,58	384256,00
5	1873,33	1,82	99,00	1880,45	370548,00
6	2034,12	1,80	99,00	2040,60	372192,00
7	1894,34	1,73	99,00	1901,32	374308,00
8	2036,58	1,78	99,00	2043,29	369776,00
9	1563,99	1,57	99,00	1570,99	377960,00
10	2812,15	2,29	99,00	2818,90	397988,00
Average	**2348,00**	**1,96**	**99,00**	**2354,47**	**379298,40**
Standard deviation	**697,79**	**0,30**	**0,00**	**697,12**	**11521,13**

With an average wall clock time of 40 minutes per training session, the Neural Network proved to be computationally intensive. This metric is also inconsistent, with a standard deviation of over ten minutes. The maximum resident size was much more conservative, at under 400 Megabytes for the majority of runs, with a standard deviation of just over 10 Megabytes.

Each run saturated the single thread performance of the processor, consistently reporting a 99% CPU time dedication. This confirms that, given the current hardware specifications, training a Neural Network is bounded by the performance of the CPU, not the I/O. Any further performance gains would require a faster clock speed on a single thread.

The overwhelming majority (99.9% on average) of CPU time was spent in user space, indicating only minor overhead for system calls during training.

6.2.4.2 Evaluation metrics

The trained classifier was then cross validated using 20 000 Tweets remaining from each split. The results of this validation are shown in Table 6.33.

Table 6.33: Neural Network evaluation metrics

Experiment #	Precision	Recall	F-measure	Accuracy %
1	0,89	0,89	0,89	89,22%
2	0,89	0,89	0,89	89,25%
3	0,89	0,89	0,89	88,92%
4	0,89	0,89	0,89	88,58%
5	0,89	0,89	0,89	89,05%
6	0,89	0,89	0,89	88,92%
7	0,89	0,89	0,89	89,29%
8	0,89	0,89	0,89	89,07%
9	0,89	0,89	0,89	89,01%
10	0,89	0,89	0,89	89,11%
Average	**0,89**	**0,89**	**0,89**	**89,04%**
Standard deviation	**0,00**	**0,00**	**0,00**	**0,21%**

The results show that the performance of the Neural Network was highly consistent, with only minor variations in accuracy. Differences in precision, recall and F-measure were negligible. The seventh run attained the highest accuracy, at 89.29%.

6.2.4.3 Testing metrics

During evaluation, the seventh classifier displayed the highest accuracy and was therefore chosen for testing. The classifier was then applied to 25 000 Tweets on a cluster size of between three and eight machines, with ten runs completed at each cluster size.

Table 6.34: Neural Network performance metrics for three machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,334250	0,000024	0,001726	880,7
2	0,330513	0,000025	0,001563	880,7
3	0,330184	0,000024	0,001448	880,6
4	0,336856	0,000025	0,001451	880,7
5	0,336556	0,000026	0,001611	880,6
6	0,331572	0,000024	0,001825	880,6
7	0,326914	0,000024	0,001482	880,7
8	0,330376	0,000025	0,001514	880,7
9	0,330406	0,000025	0,001428	880,7
10	0,335793	0,000027	0,001543	880,7
Average	**0,332342**	**0,000025**	**0,001559**	**880,7**

With the participation of three machines, the classifier displays consistent performance across all ten runs, with very little variation in any of the four metrics measured.

Table 6.35: Neural Network performance metrics for four machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,335762	0,000026	0,001680	881,0
2	0,331187	0,000025	0,001778	881,0
3	0,331627	0,000025	0,001571	881,0
4	0,335444	0,000027	0,001699	881,0
5	0,333679	0,000028	0,001671	881,0
6	0,327298	0,000024	0,001585	881,0
7	0,334772	0,000026	0,001646	881,0
8	0,334559	0,000027	0,001714	881,0
9	0,335123	0,000026	0,001539	881,0
10	0,334021	0,000026	0,001681	881,0
Average	**0,333347**	**0,000026**	**0,001657**	**881,0**

The addition of a fourth machine results in a very slight average increase in classification duration and database time, and a minor decrease in CPU time compared to the results of three machines. The results remain fairly consistent.

Table 6.36: Neural Network performance metrics for five machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,338476	0,000025	0,001740	882,1
2	0,340682	0,000026	0,001647	882,1
3	0,337037	0,000025	0,001612	882,1
4	0,344634	0,000025	0,001730	882,1
5	0,342832	0,000026	0,001776	882,1
6	0,338748	0,000024	0,001791	882,1
7	0,334239	0,000024	0,001608	882,1
8	0,336053	0,000023	0,001610	882,1
9	0,333941	0,000023	0,001599	882,2
10	0,335168	0,000023	0,001841	882,1
Average	**0,338181**	**0,000024**	**0,001696**	**882,1**

Performance and resource usage remain consistent with the addition of a fifth machine. The results are comparatively similar to the four-machine cluster.

Table 6.37: Neural Network performance metrics for six machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,341836	0,000029	0,002001	881,6
2	0,334315	0,000025	0,001885	881,6
3	0,335622	0,000025	0,001917	881,6
4	0,337222	0,000028	0,002204	881,6
5	0,332167	0,000025	0,001826	881,6
6	0,332161	0,000027	0,001760	881,6
7	0,336686	0,000027	0,001822	881,6
8	0,336950	0,000027	0,001679	881,6
9	0,339592	0,000027	0,001793	881,6
10	0,331514	0,000026	0,001856	881,6
Average	**0,335806**	**0,000027**	**0,001874**	**881,6**

The addition of the sixth machine results in a slight increase in average database time compared to the five-machine cluster, with all metrics remaining consistent across the ten runs.

Table 6.38: Neural Network performance metrics for seven machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,337939	0,000024	0,001732	882,0
2	0,338194	0,000023	0,001796	882,0
3	0,340013	0,000024	0,001808	882,0
4	0,341785	0,000025	0,001709	882,0
5	0,341531	0,000024	0,001737	882,0
6	0,341171	0,000024	0,001835	882,0
7	0,343558	0,000025	0,001818	882,0
8	0,342747	0,000026	0,001760	882,0
9	0,331177	0,000023	0,001566	882,0
10	0,339001	0,000024	0,001826	882,0
Average	**0,339712**	**0,000024**	**0,001759**	**882,0**

The seven-machine cluster continues the trend of consistent performance, and compares favourably with the six-machine cluster, with slightly better results on average across all metrics.

Table 6.39: Neural Network performance metrics for eight machines

Experiment #	CPU time (s)	Duration (s)	Database time (s)	Max RS (MB)
1	0,330865	0,000023	0,002878	882,0
2	0,332706	0,000024	0,002077	882,0
3	0,334522	0,000022	0,001836	882,0
4	0,334769	0,000024	0,001765	882,0
5	0,340003	0,000027	0,001804	882,0
6	0,339468	0,000025	0,001824	882,0
7	0,338430	0,000024	0,001861	882,0
8	0,337197	0,000025	0,001744	882,0
9	0,336952	0,000023	0,001705	882,0
10	0,336926	0,000024	0,001774	882,0
Average	**0,336184**	**0,000024**	**0,001927**	**882,0**

Finally, the addition of the eighth machine concludes with similarly consistent performance across all ten runs, with a slight increase in CPU time, database time and resident size, combined with a minor reduction in classification duration.

A summary of the average values for each metric measured at each machine count is shown in Table 6.40 below:

Table 6.40: Average Neural Network performance metrics for all machine counts

Machine count	CPU time	Duration	Database time	Max RS (KB)
3	0,332342	0,000025	0,001559	880,7
4	0,333347	0,000026	0,001657	881,0
5	0,338181	0,000024	0,001696	882,1
6	0,335806	0,000027	0,001874	881,6
7	0,339712	0,000024	0,001759	882,0
8	0,336184	0,000024	0,001927	882,0

These figures can be graphically represented as a chart for each metric:

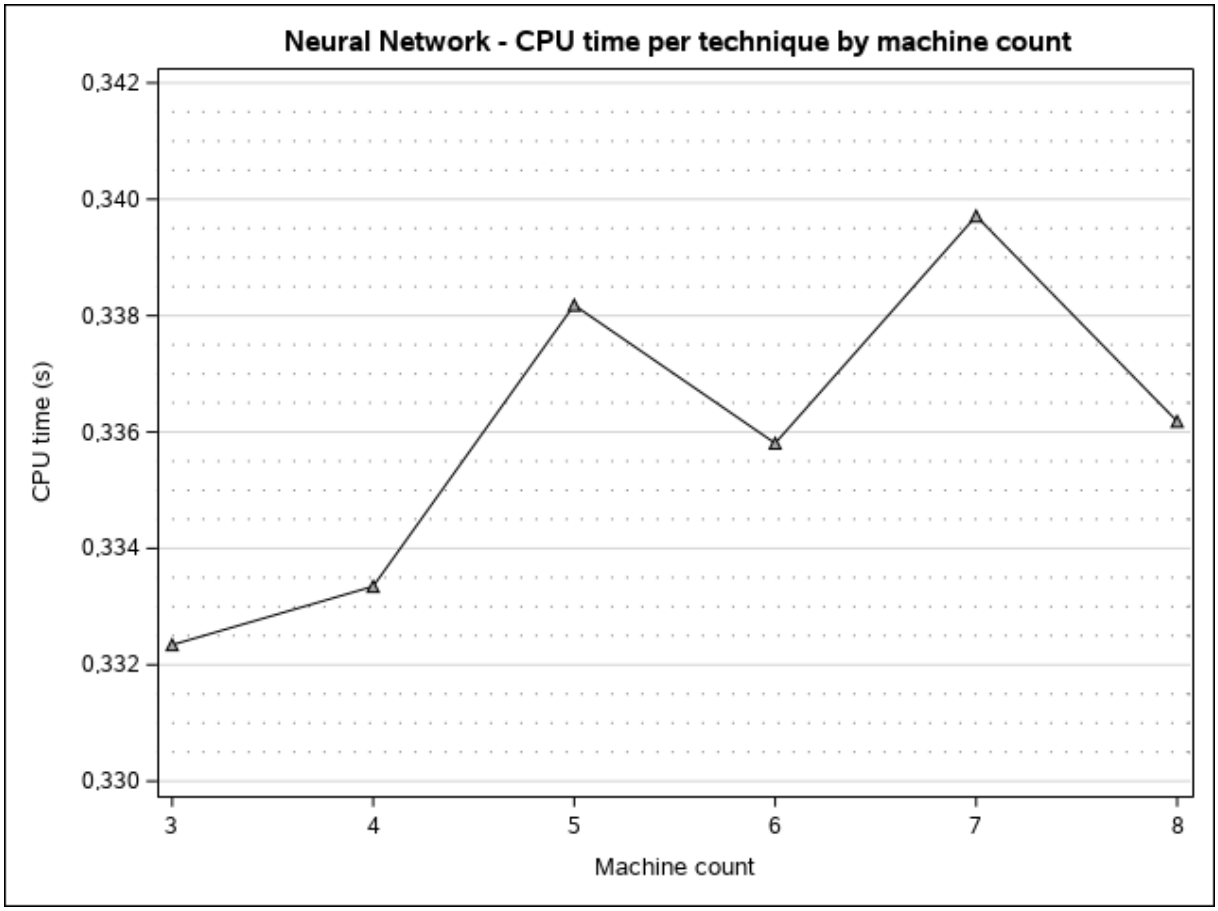

Figure 6.13: Average Neural Network CPU time by machine count

The Neural Network classifier displays a general increase in CPU time as the number of machines are increased, as shown in Figure 6.13, with a variation of between 0,002 and 0,003 seconds disrupting the pattern.

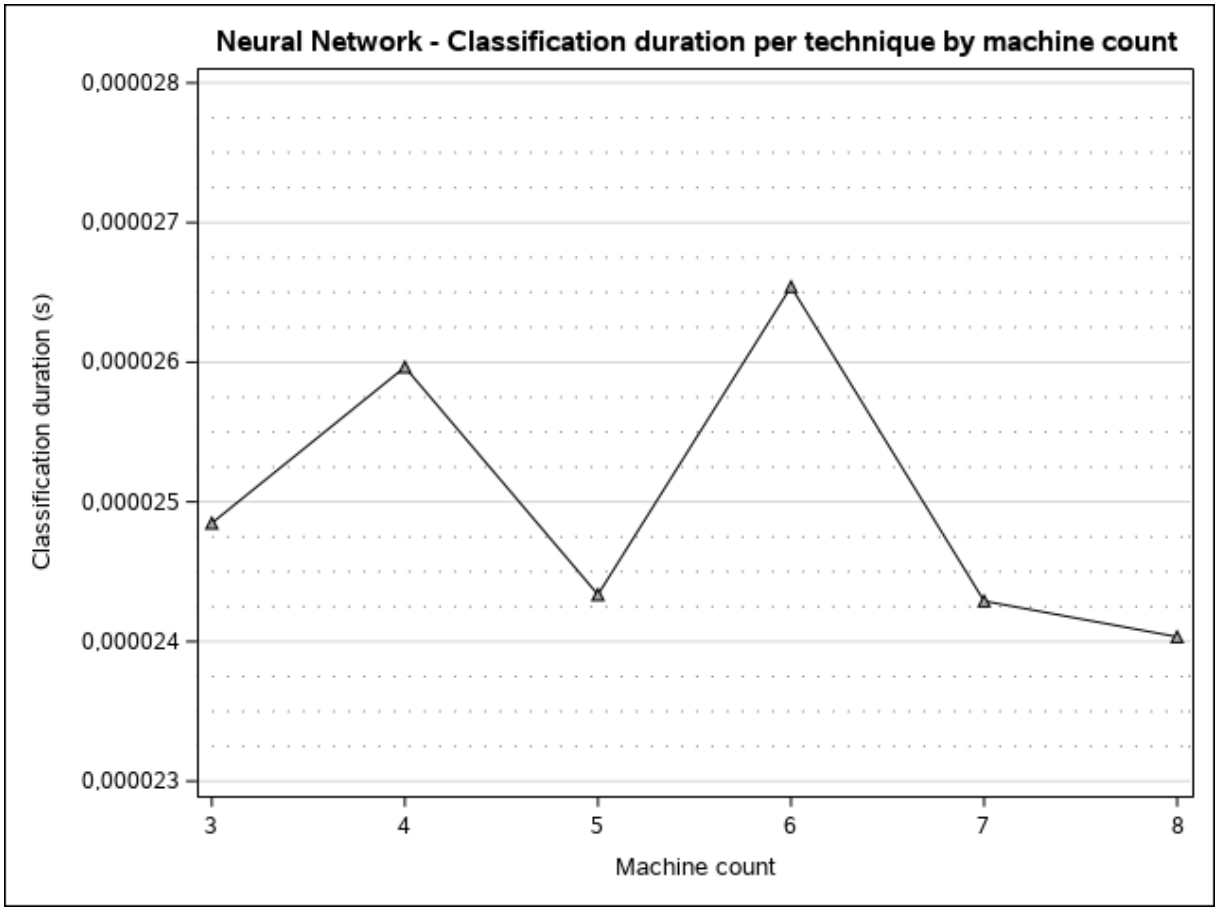

Figure 6.14: Average Neural Network duration by machine count

The average classification duration for the Neural Network classifier, shown in Figure 6.14 does not display any particular pattern of increase or reduction as the number of machines are increased.

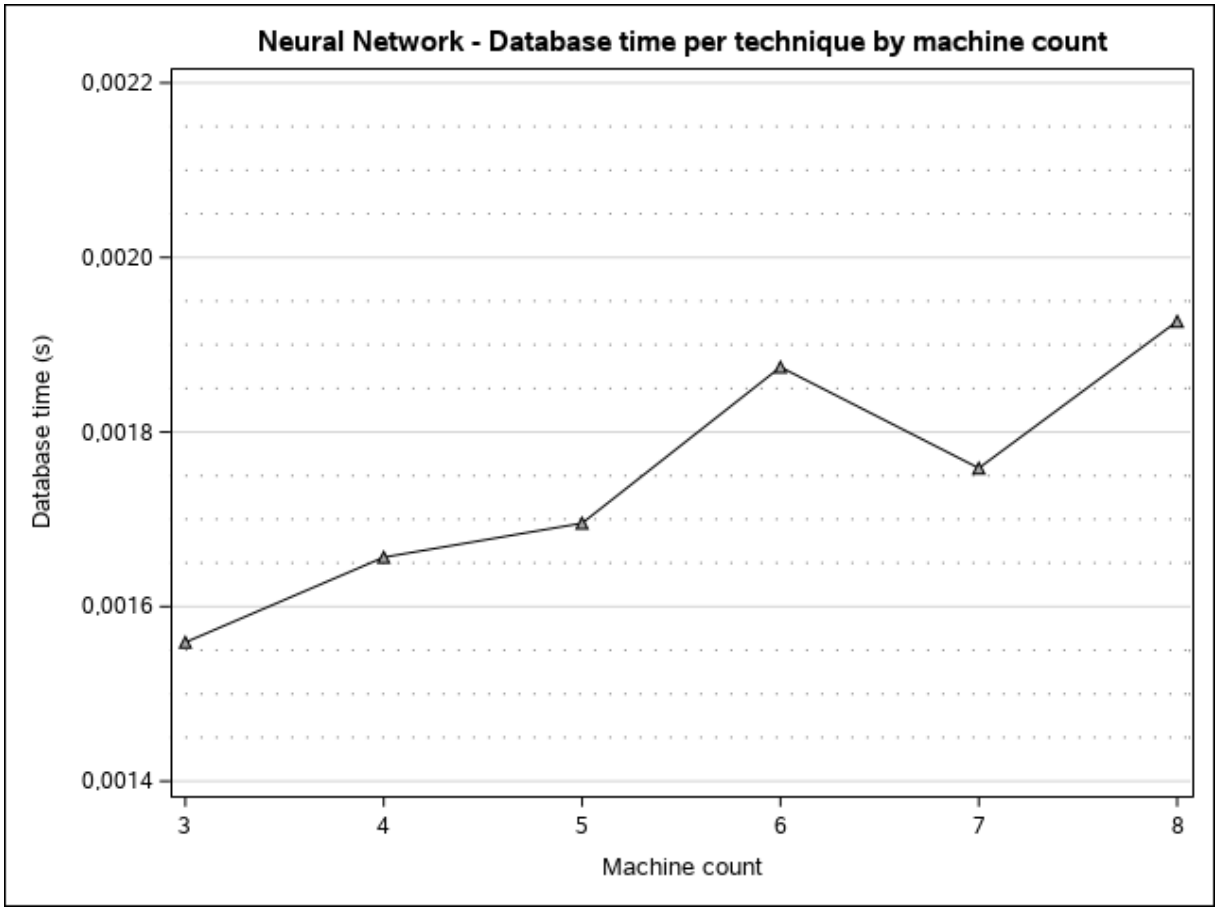

Figure 6.15: Average Neural Network database time by machine count

The Neural Network classifier displays a more gradual increase in database time compared to the previous classifiers, showing a steady increase as the number of machines rises.

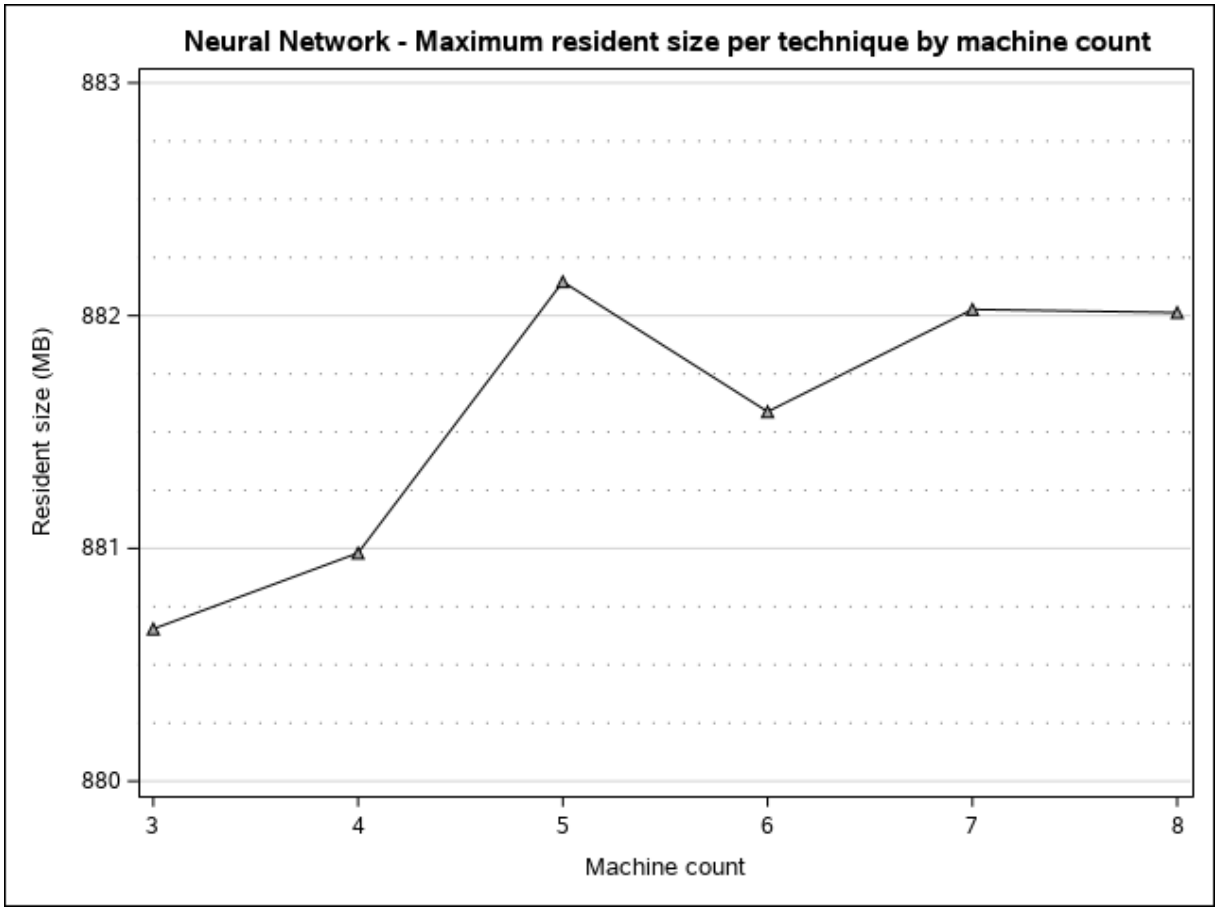

Figure 6.16: Average Neural Network maximum resident size by machine count

The average maximum resident size of the Neural Network classifier displays a slight increase as more machines are added to experiment.

Summarised classifier performance metrics for the ten runs are given in Table 6.41 below.

Table 6.41: Neural Network classifier performance metrics

Run #	False Positives	False Negatives	True Positives	True Negatives
1	322	544	10990	13144
2	322	544	10990	13144
3	322	544	10990	13144
4	322	544	10990	13144
5	322	544	10990	13144
6	322	544	10990	13144
7	322	544	10990	13144
8	322	544	10990	13144
9	322	544	10990	13144
10	322	544	10990	13144

These results indicate that the trained classifier's classification is deterministic (given the same input, it will produce the same output), since the results are identical across all ten runs and all six machine counts. The calculated values for precision, recall, F-measure and accuracy are given in Table 6.42.

Table 6.42: Summary of Neural Network classifier performance metrics

Precision	Recall	F-Measure	Accuracy
0.971535	0.960257	0.965863	0.96536

Very high precision and recall indicate a highly accurate classifier. These values combined lead to an F-measure of approximately 96.6%, and an overall accuracy of 96.5%.

6.3 Overall comparison of training metrics

This subsection evaluates the three machine learning techniques comparatively (the lexicon-based classifier has been excluded, since it does not include a training aspect). All measurements for the classifiers are the average measurements across all ten training runs documented previously.

6.3.1 Training time

Training time was measured according to user space time (denoted as usr time), system time (denoted as sys time) and total duration. A summary of these results is given in Table 6.43 below:

Table 6.43: Summary of training time by classifier

Classifier	Usr time (s)	Sys time (s)	Elapsed seconds
Naïve-Bayes	8,04	0,56	11,81
Support Vector Machine	921,24	1,15	925,5
Neural Network	2348	1,96	2354,47

These results can be illustrated graphically, as shown in Figure 6.17. Note the logarithmic scale on the y-axis.

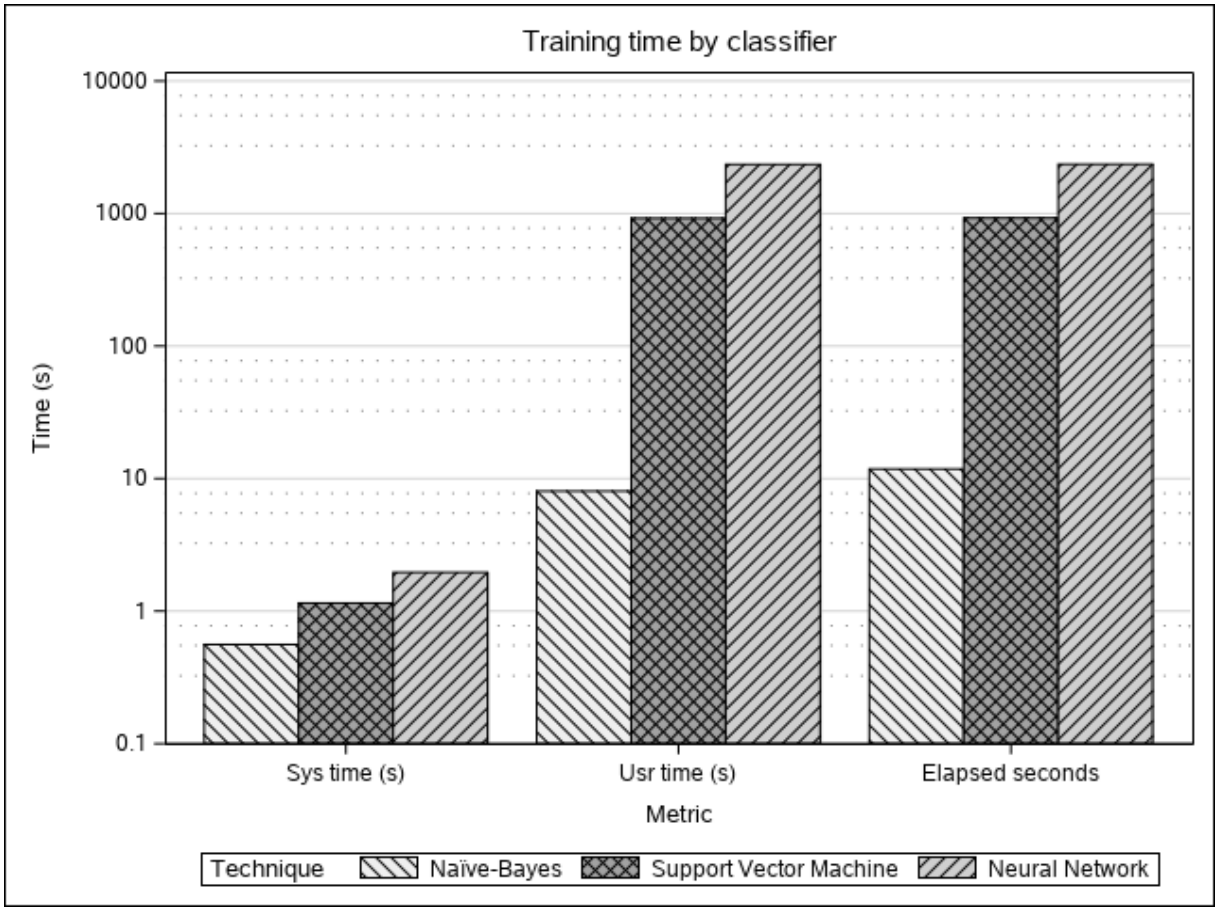

Figure 6.17: Training time by classifier

From the results it is clear that, on average, the Naïve-Bayesian classifier is considerably faster to train than the other two classifiers, at just under twelve seconds. The support vector machine, in turn, could be trained in less than half the time required to train the Neural Network classifier, with training times of fifteen and thirty-nine minutes respectively.

Such a considerable difference between the training times indicate that this is an important consideration if the quantity of training data is known in advance, since an increase in the size of the training set will result in a corresponding increase in training time.

In all cases the user time represented the majority of the training time, indicating that the system calls involved in the training procedures were minimal. The combined user and system times correspond roughly to the total duration in each case, although more so for the support vector machine and Neural Network than the Naïve-Bayesian classifier. This indicates heavy use of a single CPU core, since the use of multiple cores would typically lead to a combined user and system time that is higher than the wall-clock duration. This is confirmed in the next section.

6.3.2 CPU usage

The average CPU usage was measured as a percentage, with 100% being the maximum per core, and 400% being the overall maximum (given a machine with four cores). The average CPU usage for each classifier during training is given in Table 6.44 below.

Table 6.44: Summary of CPU usage by classifier

Classifier	CPU usage (%)
Naïve-Bayes	72
Support Vector Machine	99
Neural Network	99

These results can be illustrated graphically, as shown in Figure 6.18 below.

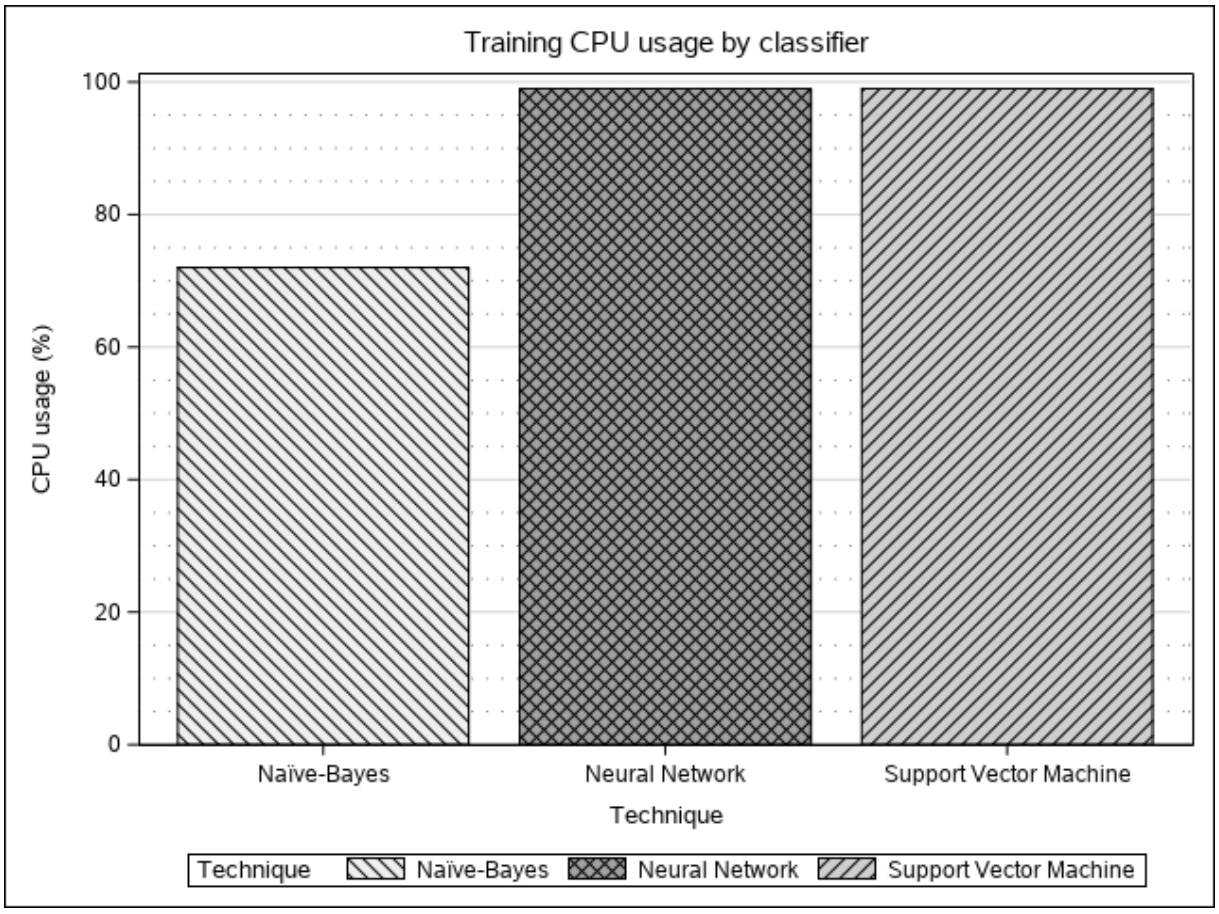

Figure 6.18: Training CPU usage by classifier

These results correspond with the results obtained in the previous section based on the relationship between the combination of user time and system time with the total duration. From these figures, it is clear that both the support vector machine and Neural Networks

saturate one CPU core during training, indicating that these are single-threaded operations as implemented in the Scikit-learn library. The Naïve-Bayesian classifier, on the other hand, did not even saturate a single CPU core. For that to be the case, it is likely that the classifier trained faster than the I/O was capable of providing data. To speed up the Naïve-Bayesian training one would therefore likely need faster I/O, instead of faster processing, unlike the other two classifiers, which require faster single-core processing performance.

6.3.3 Maximum resident size

The average maximum resident size was measured throughout the training of the three classifiers. These measurements are given in Table 6.45 below:

Table 6.45: Summary of maximum resident size by classifier

Classifier	**Maximum resident size (KB)**
Naïve-Bayes	321688,89
Support Vector Machine	526559,6
Neural Network	379298,4

These results can be illustrated graphically, as shown in Figure 6.19 below.

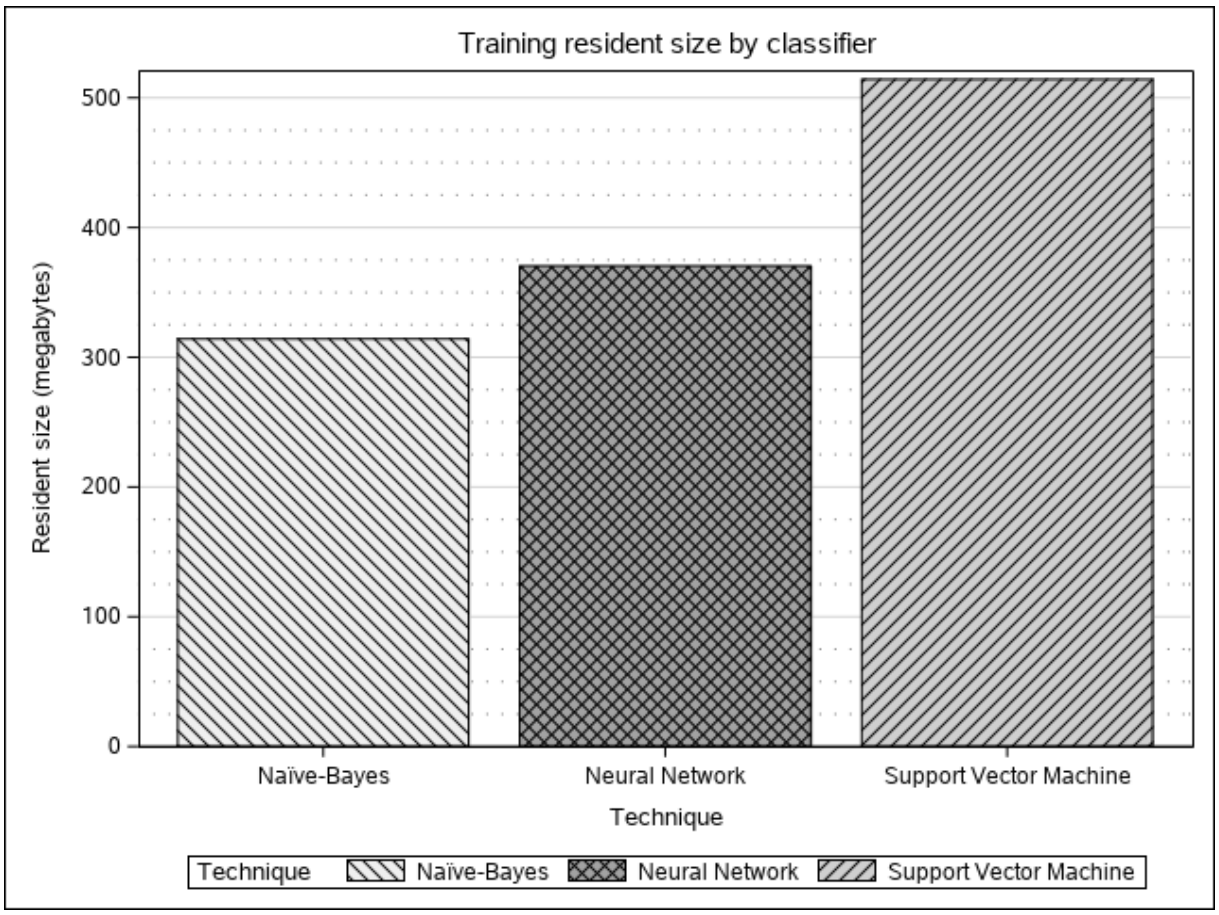

Figure 6.19: Training maximum resident size by classifier

The Naïve-Bayesian classifier is once again the least resource-intensive, occupying an average of roughly 320 megabytes of memory. The Neural Network classifier occupies slightly more during training, at roughly 380 megabytes. The Support Vector Machines proved to be the largest by far during training, occupying approximately 525 megabytes during training.

6.4 Overall comparison of testing metrics

This subsection evaluates all four techniques comparatively, using the average measurements discussed in the previous subsection.

6.4.1 CPU time

A complete overview of the CPU time consumed by each technique at every cluster size is given in Table 6.46 and shown graphically in Figure 6.20. Note the logarithmic scale on the vertical axis.

Table 6.46: CPU time per technique by machine count

Machine Count	Lexicon-based	Naïve-Bayes	SVM	Neural Network
3	0,273151	0,332877	3,545386	0,332342
4	0,274058	0,336641	3,520763	0,333347
5	0,282493	0,344981	3,530528	0,338181
6	0,280186	0,338822	3,485887	0,335806
7	0,283872	0,345904	3,520434	0,339712
8	0,282371	0,339261	3,540227	0,336184

Please note that all times are given in seconds.

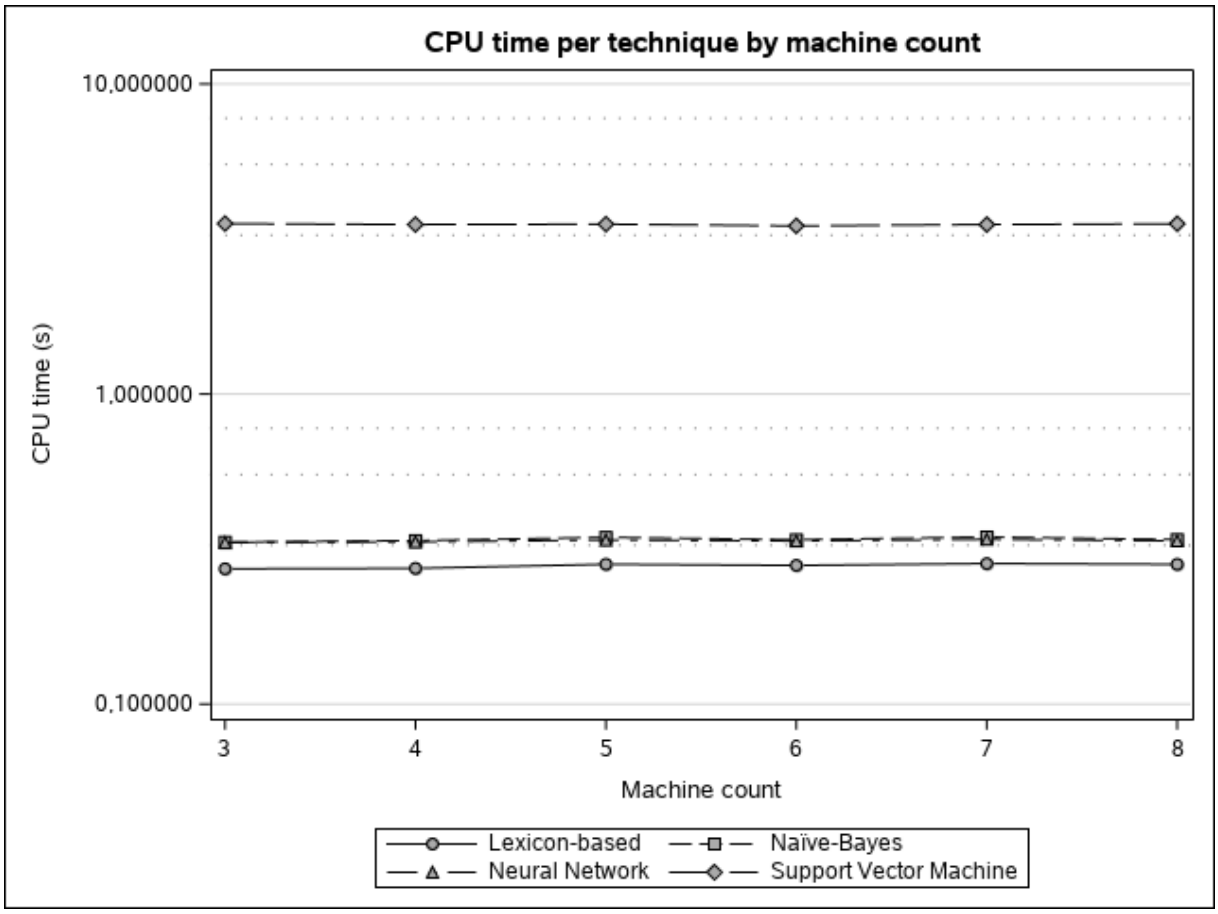

Figure 6.20: CPU time per technique by machine count

Of the four classifiers, the SVM classifier is the most CPU-intensive; more so by an order of magnitude in comparison to the other three. The Naïve-Bayes and Neural Network classifiers are very similar in terms of performance, with the Lexicon-based classifier making the least use of the processor time. It is also important to note that while the CPU time spent on the SVM classifier is high, it is still below four seconds, which would be the theoretical maximum on a

machine with four CPU cores. From this we can deduce that the remaining CPU time was likely spent on system calls and the database.

6.4.2 Classification duration

A comparative overview of the classification duration for each technique at every machine count is given in Table 6.47 below:

Table 6.47: Duration per technique by machine count

Machine Count	Lexicon-based	Naïve-Bayes	SVM	Neural Network
3	0,000021	0,000023	0,000037	0,000025
4	0,000022	0,000025	0,000038	0,000026
5	0,000022	0,000026	0,000038	0,000024
6	0,000023	0,000025	0,000041	0,000027
7	0,000021	0,000025	0,000041	0,000024
8	0,000021	0,000023	0,000040	0,000024

Please note that all times are given in seconds. These results are graphically represented in Figure 6.21.

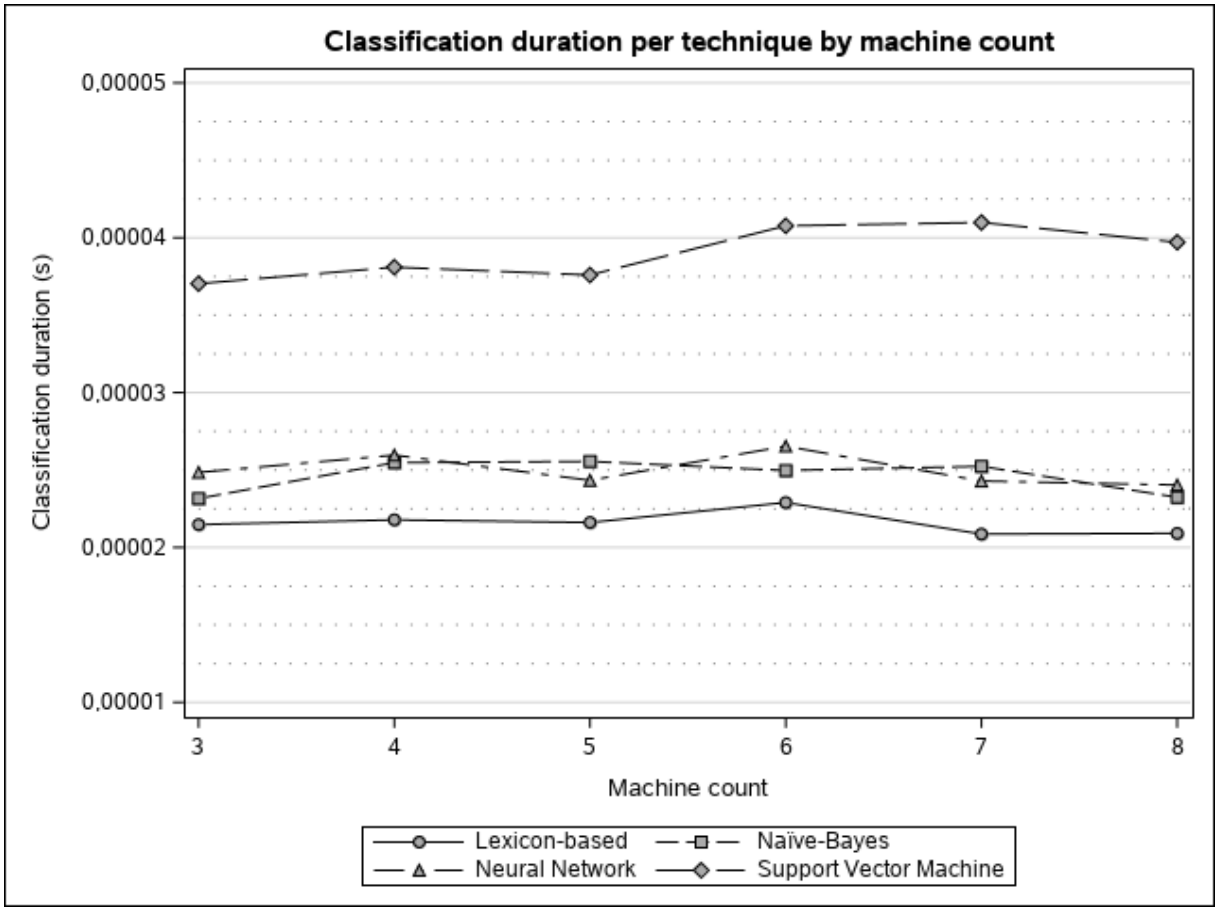

Figure 6.21: Duration per technique by machine count

The SVM classifier takes the longest time per classification, by some margin. The remaining three techniques are very similar in their performance, with the lexicon-based classifier having a slight edge in terms of performance on the remaining two classifiers. No clear pattern emerges between the different techniques as the number of machines is increased.

6.4.3 Database time

The performance of the database was measured during every test stage, at every cluster size. The time spent to deposit a result (i.e. a tweet with its attributes and sentiment classification) was recorded. The average latency per insert statement for every sentiment analysis technique at every machine count is given in Table 6.48 and graphically shown in Figure 6.22.

Table 6.48: Database time per technique by machine count

Machine Count	Lexicon-based	Naïve-Bayes	SVM	Neural Network
3	0,001438	0,001702	0,002864	0,001559
4	0,001623	0,001689	0,002962	0,001657
5	0,001604	0,001645	0,002787	0,001696
6	0,001702	0,001978	0,003300	0,001874
7	0,001682	0,001783	0,002847	0,001759
8	0,001653	0,001819	0,002884	0,001927

Please note that all times are given in seconds.

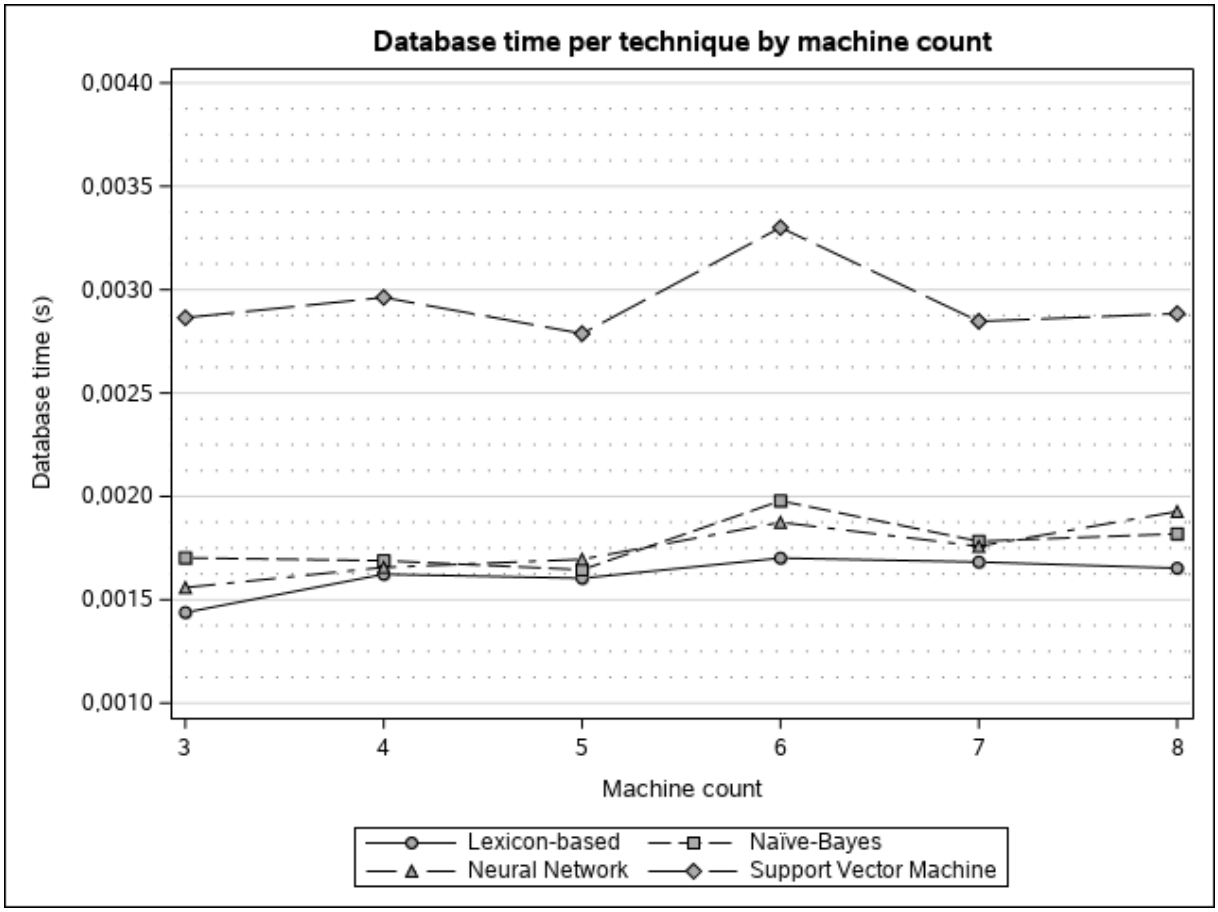

Figure 6.22: Database time per technique by machine count

Three of the classifiers display very similar results, with the exception of the SVM classifier, which exhibits a similar pattern, but with a considerable overall increase in the duration. This difference can be explained by the CPU-intensive nature of the SVM classifier, which saturates the CPU cores during classification and therefore delays the database processing which runs in parallel on the same machines as the classifiers. For the other three classifiers, the classification

process is less CPU-intensive, which would leave other CPU cores available to handle the database processing. The CPU time discussion in section 6.4.1 supports this argument, since the CPU time devoted to the classifier was very close to the theoretical maximum, leaving very little headroom for the system calls and the database.

6.4.4 Maximum resident size

The average maximum resident size for each classifier at every cluster size is given in Table 6.49.

Table 6.49: Maximum resident size per technique by machine count

Machine Count	Lexicon-based	Naïve-Bayes	SVM	Neural Network
3	818,0706	863,2908	884,6155	880,6541
4	818,5291	863,8974	885,2339	880,9801
5	819,5431	864,7927	886,1136	882,1455
6	819,1495	864,2262	885,5665	881,5875
7	819,6134	864,9216	886,2476	882,0266
8	819,3209	864,6453	885,9772	882,0130

Table measurements are given in bytes. These metrics are shown graphically in Figure 6.23. Graph measurements are given in megabytes.

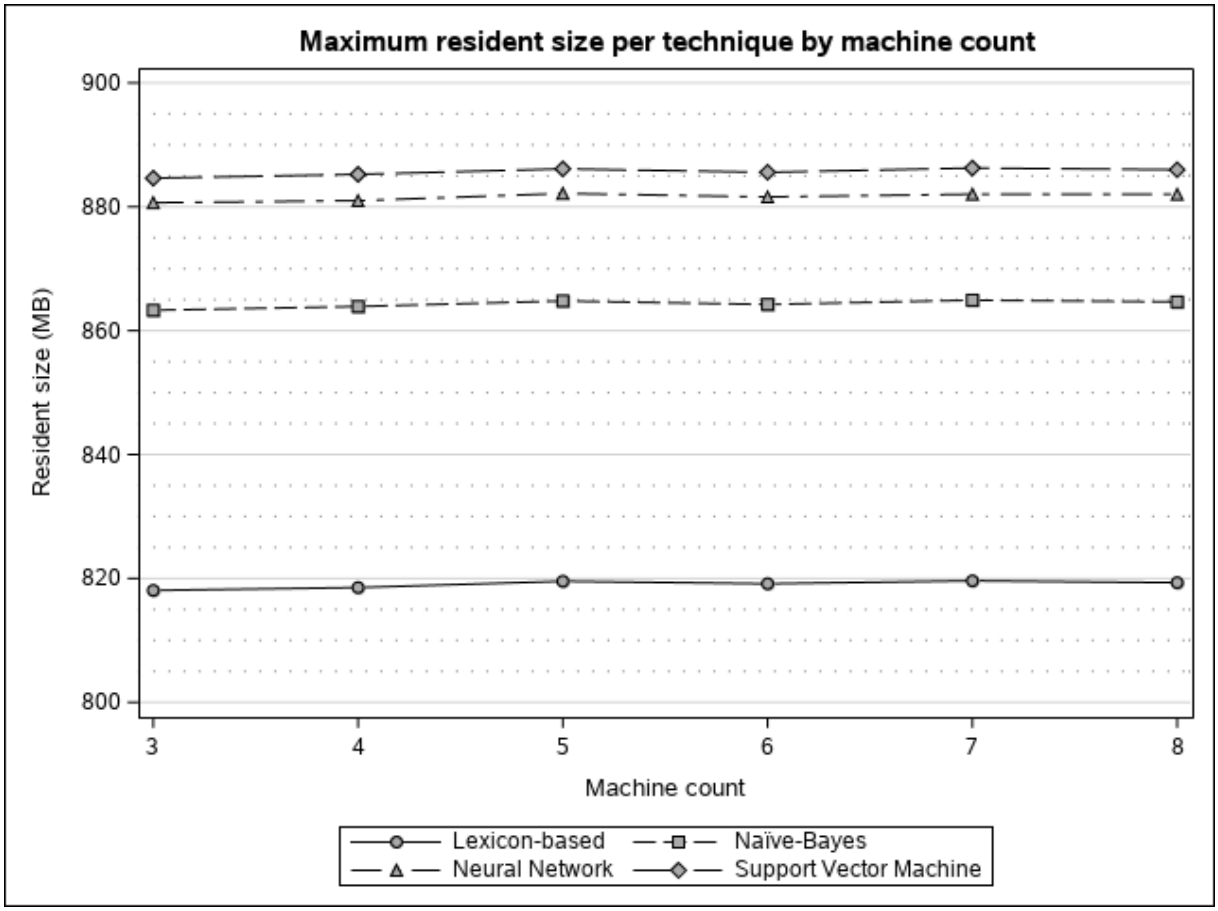

Figure 6.23: Maximum resident size per technique by machine count

The SVM and Neural network classifiers occupy the most memory across the board, with the SVM classifier using about five megabytes more on average. The Naïve-Bayes classifier comes in at a close third, using about twenty megabytes less on average. The lightest of the four is the lexicon-based classifier, consistently utilising just under 820 megabytes.

6.4.5 Classifier performance metrics

All classifiers behaved deterministically, so in this case the accuracy of the classifiers can be compared irrespective of the number of machines partaking in the experiment. Note that all four classifiers were tested using binary classes, i.e. positive and negative classes. The classifier performance metrics statistics for the four classifiers obtained under the test dataset are given in Table 6.50.

Table 6.50: Overall classifier performance

Classifier	Precision	Recall	F-Measure	Accuracy
Lexicon-based	0,6510	0,6329	0,6418	0,6741
Naïve-Bayes	0,9222	0,7018	0,7971	0,8351
Support Vector Machine	0,9410	0,5708	0,7106	0,7855
Neural Network	0,9715	0,9528	0,9621	0,9654

Bear in mind that these metrics are determined according to the calculations discussed in more detail in section 2.6.2.4. Precision and recall were calculated according to the number of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn):

$$precision = \frac{tp}{tp + fp} \tag{9}$$

$$recall = \frac{tp}{tp + fn} \tag{10}$$

These two elements were then combined into a single metric known as the F_1-measure (or F_1-score). The relationship between the F_1-measure and precision and recall variables is shown in the equation below:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{11}$$

This data can be represented graphically, as show in Figure 6.21.

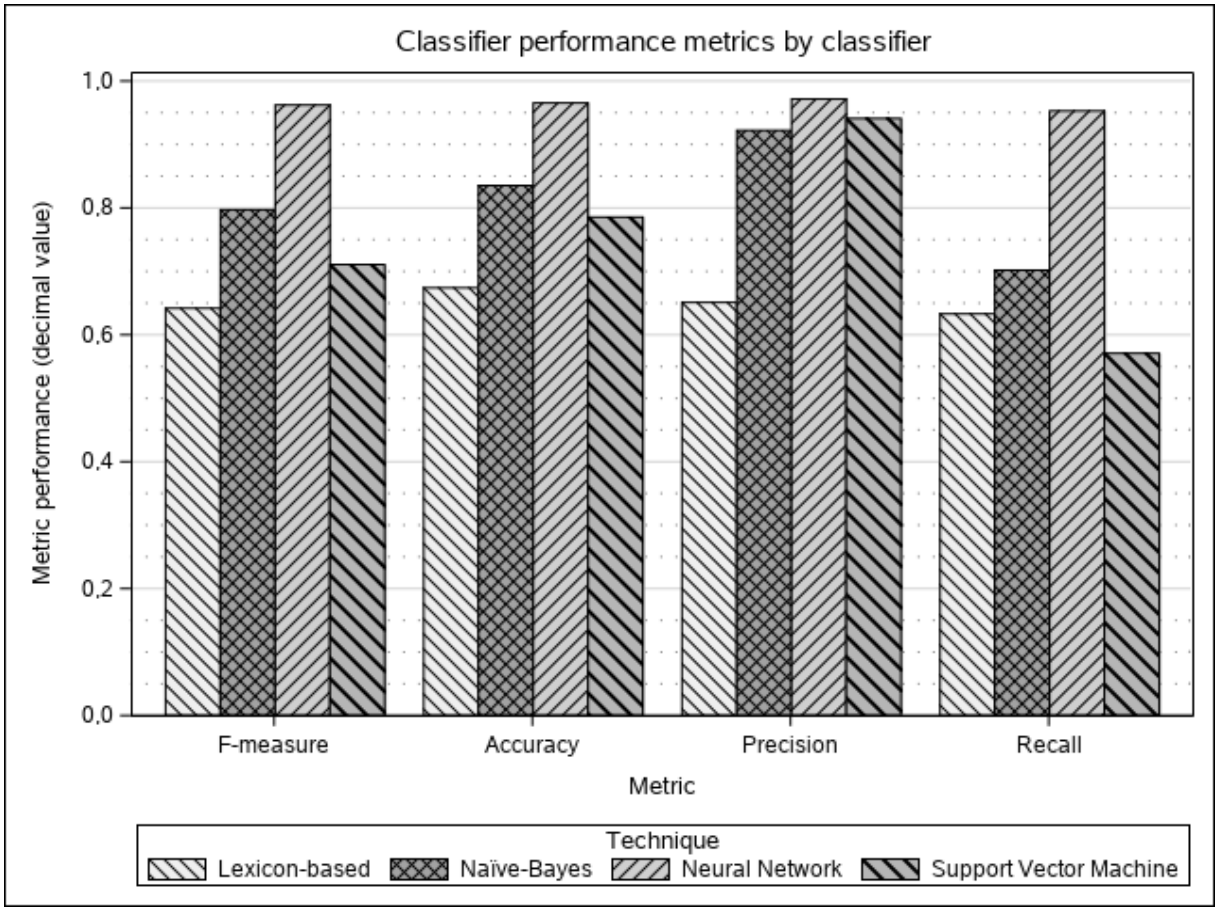

Figure 6.24: Accuracy metrics by classifier

The Neural Network classifier leads by some margin in almost all cases. In the case of precision, the Naïve-Bayes and SVM classifiers come close, exceeding 0.9, but the recall values are much lower, at around 0.6 to 0.7, reducing the overall F-measure and accuracy for both classifiers quite considerably. The Lexicon-based classifier was by far the worst, with an overall accuracy of only 67%, and similarly poor results in terms of precision and recall. It is, however, the fastest and least resource-intensive compared to the other three classifiers. Interestingly, the SVM classifier, which used noticeably more resources than the Neural Network classifier performed worse in comparison. The resource usage between the Naïve-Bayes classifier was similar to that of the Neural Network classifier, but the Neural Network classifier provided better results in all cases.

6.4.6 Hypothesis testing

To test the null hypothesis that the predictive performance of models is equal (using a significance level of α=0.05), we can conduct a corrected McNemar Test for computing the chi-squared and p-values. The McNemar Test relies on McNemar contingency matrices, which

in turn makes use of the confusion matrixes for each classifier. The confusion matrices for each of the classifiers are as follows:

Table 6.51: Lexicon-based classifier confusion matrix

n = 25 000	**Predicted: No**	**Predicted: Yes**
Actual: No	9552	3914
Actual: Yes	4234	7300

Table 6.52: Naïve-Bayes classifier confusion matrix

n = 25 000	**Predicted: No**	**Predicted: Yes**
Actual: No	12783	683
Actual: Yes	3439	8095

Table 6.53: Support Vector Machine classifier confusion matrix

n = 25 000	**Predicted: No**	**Predicted: Yes**
Actual: No	13053	413
Actual: Yes	4950	6584

Table 6.54: Neural Network classifier confusion matrix

n = 25 000	**Predicted: No**	**Predicted: Yes**
Actual: No	13144	322
Actual: Yes	544	10990

These confusion matrices allow for a summarised report of each classifier's performance at a glance, and also intuitively illustrates the precision and recall attributes.

Starting with McNemar contingency matrices for each combination of classifier, we can determine if there are differences in their classification performance and then make use of McNemar tests to determine whether these differences are statistically significant. The following sections will test all our null hypotheses to determine of the predictive performance of two models are equal (using a significance level of α=0.05).

6.4.6.1 Lexicon-based and Naïve-Bayes

$H_{0,1}$: There is no significant difference between the predictive performance of the lexicon-based and the Naïve-Bayes models.

The McNemar contingency matrix for the lexicon-based and Naïve-Bayes classifiers is shown in Table 6.55 below:

Table 6.55: Lexicon-based and Naïve-Bayes classifiers McNemar contingency matrix

	Naïve-Bayes correct	**Naïve-Bayes incorrect**
Lexicon-based correct	15048	1804
Lexicon-based incorrect	5830	2318

From this table it is clear that the Naïve-Bayes classifier outperforms the lexicon-based classifier in terms of correct predictions, with a ratio of approximately 1:3 (1804 versus 5830) in terms of incorrect predictions unique to each classifier. The results of the McNemar Test are as follows:

Chi-square value: 2122.17
P-value: $< 2.2250738585072014 \times 10^{-308}$ (reported as 0.00)

> Note that the threshold for the p-value shown above is the minimum possible floating-point value that Python can represent above zero. We can, therefore, report that the p-value is less than this figure, but likely not zero. It is, however, only important that the value can be determined as either larger or smaller than α in order to proceed with hypotheses testing.

Since the p-value is smaller than our assumed significance threshold (α=0.05), we **can** reject our null-hypothesis and assume that there is a significant difference between the two predictive models.

6.4.6.2 Lexicon-based and Support Vector Machine

$H_{0,2}$: There is no significant difference between the predictive performance of the lexicon-based and the Support Vector Machine models.

The McNemar contingency matrix for the lexicon-based and Support Vector Machine classifiers is given in Table 6.56 below:

Table 6.56: Lexicon-based and Support Vector Machine classifiers McNemar contingency matrix

	Support Vector Machine correct	**Support Vector Machine incorrect**
Lexicon-based correct	14510	2342
Lexicon-based incorrect	5127	3021

This table indicates that the Support Vector Machine outperforms the lexicon-based classifier by a ratio of about 1:2 (2342 versus 5127) in terms of incorrect predictions unique to each classifier. The results of the McNemar Test are as follows:

Chi-square value: 1037.71
P-value: $3.358538682084105 \times 10^{-233}$

Since the p-value is smaller than our assumed significance threshold (α=0.05), we **can** reject our null-hypothesis and assume that there is a significant difference between the two predictive models.

6.4.6.3 Lexicon-based and Neural Network

$H_{0,3}$: There is no significant difference between the predictive performance of the lexicon-based and Neural Network models.

The contingency table for the lexicon-based and Neural Network classifiers is shown in Table 6.57 below:

Table 6.57: Lexicon-based and Neural Network classifiers McNemar contingency matrix

	Neural Network correct	**Neural Network incorrect**
Lexicon-based correct	16476	376
Lexicon-based incorrect	2342	490

This indicates that the Neural Network classifier considerably outpaces the lexicon-based classifier. The lexicon-based classifier had almost eight times (376 versus 2342) the number of incorrect predictions compared to the Neural Network, in terms of incorrect predictions unique to each classifier. The results of the McNemar Test are as follows:

Chi-square value: 6598.58
P-value: < 2.2250738585072014 x 10^{-308} (reported as 0.00)

Since the p-value is smaller than our assumed significance threshold (α=0.05), we **can** reject our null-hypothesis and assume that there is a significant difference between the two predictive models.

6.4.6.4 Naïve-Bayes and Support Vector Machine

$H_{0,4}$: There is no significant difference between the predictive performance of the Naïve-Bayes and Support Vector Machine models.

The McNemar contingency matrix for the Naïve-Bayes and Support Vector Machine classifiers is given in Table 6.58 below:

Table 6.58: Naïve-Bayes and Support Vector Machine classifiers McNemar contingency matrix

	Support Vector Machine correct	**Support Vector Machine incorrect**
Naïve-Bayes correct	19001	636
Naïve-Bayes incorrect	1877	3486

From this table it is clear that the Support Vector Machine outperforms the Naïve-Bayes classifier by roughly 1:3 (636 versus 1877) in terms of unique incorrect predictions. The results of the McNemar Test are as follows:

Chi-square value: 611.86
P-value: 4.4363794382758815 x 10^{-141}

Since the p-value is smaller than our assumed significance threshold (α=0.05), we **can** reject our null-hypothesis and assume that there is a significant difference between the two predictive models.

6.4.6.5 Naïve-Bayes and Neural Network

$H_{0,5}$: There is no significant difference between the predictive performance of the Naïve-Bayes and Neural Network models.

The contingency matrix for the Naïve-Bayes and Neural Network classifiers is shown in Table 6.59 below:

Table 6.59: Naïve-Bayes and Neural Network classifiers McNemar contingency matrix

	Neural Network correct	**Neural Network incorrect**
Naïve-Bayes correct	20550	328
Naïve-Bayes incorrect	3584	538

From the table it is clear that the performance of the Neural Network far exceeds that of the Naïve-Bayes classifier. The difference in terms of unique incorrect predictions results in a ratio of 1:10 (328 versus 3584) in favour of the Neural Network. The results of the McNemar Test are as follows:

Chi-square value: 2708.34
P-value: $< 2.2250738585072014 \times 10^{-308}$ (reported as 0.00)

Since the p-value is smaller than our assumed significance threshold (α=0.05), we **can** reject our null-hypothesis and assume that there is a significant difference between the two predictive models.

6.4.6.6 Support Vector Machine and Neural Network

$H_{0,6}$: There is no significant difference between the predictive performance of the Support Vector Machine and Neural Network models.

The McNemar contingency matrix for the Support Vector Machine and Neural Network classifiers is shown in Table 6.60 below:

Table 6.60: Support Vector Machine and Neural Network classifiers McNemar contingency matrix

	Neural Network correct	**Neural Network incorrect**
Support Vector Machine correct	19331	306
Support Vector Machine incorrect	4803	560

Based on this table it is clear that the Neural Network considerably outperforms the Support Vector Machine, with a difference of 1:16 (306 versus 4803) in terms of unique incorrect predictions. The results of the McNemar Test are as follows:

Chi-square value: 3956.55
P-value: < 2.2250738585072014 x 10^{-308} (reported as 0.00)

Since the p-value is smaller than our assumed significance threshold (α=0.05), we **can** reject our null hypothesis and assume that there is a significant difference between the two predictive models.

6.4.6.7 Cochran's Q Test for all classifiers

$H_{0,7}$: There is no significant difference between the predictive performance of the four models.

To test this hypothesis, we will make use of Cochran's Q test to compare two or more models with each other. The rest will calculate the Q (chi-squared) value as well as the *p*-value. The results of the Cochran's Q Test are as follows:

Q-value: 9462.41
P-value: < 2.2250738585072014 x 10^{-308} (reported as 0.00)

Assuming that we conducted this test also with a significance level of α=0.05, we **can** reject the null-hypothesis that all the models perform equally well on this dataset, since the *p*-value is smaller than α.

6.4.6.8 Summary of hypotheses

A summary of the hypotheses with their calculated p-values and their acceptance or rejection is indicated in Table 6.61 below.

Table 6.61: Summary of hypotheses

Hypothesis	***p*-value (*)**	**Reject or Accept (**)**
$H_{0,1}$: There is no significant difference between the predictive performance of the lexicon-based and the Naïve-Bayes models.	< 2,2250738585072014 x 10^{-308}	Reject
$H_{0,2}$: There is no significant difference between the predictive performance of the lexicon-based and the Support Vector Machine models.	3,358538682084105 x 10^{-233}	Reject
$H_{0,3}$: There is no significant difference between the predictive performance of the lexicon-based and Neural Network models.	< 2,2250738585072014 x 10^{-308}	Reject
$H_{0,4}$: There is no significant difference between the predictive performance of the Naïve-Bayes and Support Vector Machine models.	4,4363794382758815 x 10^{-141}	Reject
$H_{0,5}$: There is no significant difference between the predictive performance of the Naïve-Bayes and Neural Network models.	< 2,2250738585072014 x 10^{-308}	Reject
$H_{0,6}$: There is no significant difference between the predictive performance of the Support Vector Machine and Neural Network models.	< 2,2250738585072014 x 10^{-308}	Reject
$H_{0,7}$: There is no significant difference between the predictive performance of all four models.	< 2,2250738585072014 x 10^{-308}	Reject

(*) 2,2250738585072014 x 10^{-308} is the minimum float value for Python, reported as zero during calculation

(**) significance level of α=0,05

It was found that the null hypotheses could be rejected in every case, indicating that all classifiers performed significantly differently as measured by the McNemar and Cochran's Q-tests.

6.5 Summary

This chapter reported on the results obtained from the experimentation, as well as a detailed discussion thereof. This discussion included an analysis of all four classifiers in terms of both resource usage and classifier performance metrics, both during training and testing. These reports covered the performance of each classifier at every cluster size, from three to eight machines, and provided an overview of the changes measured as a function of the addition of more machines. This was followed by the testing of the hypotheses stated earlier in the study

In the case of classifier performance metrics, it was found that there were statistically significant differences between all four classifiers in terms of McNemar tests and Cochran's Q-tests.

The next chapter will conclude the dissertation with a brief summary of the dissertation, which includes the discussion of the research questions and how each can be answered with the results obtained during experimentation. A discussion of the results broken-down as a series of reflections on the study as a whole is given, followed by the limitations of the study. The chapter will conclude with the recommendations made for policy and practice, possible future research, and further development work.

CHAPTER 7
DISCUSSION AND CONCLUSION

7.1 Introduction

Chapter 6 presented the results of the experimentation with a detailed discussion of each individual experiment as well as the outcomes of the research project as a whole. This included a complete breakdown of the results by sentiment analysis approach and metric, prior to an overall comparison for training and testing metrics. This was followed by the testing of the hypotheses stated earlier in the study.

This chapter will summarise the dissertation and discuss the results by revisting the research questions posed in Chapter 1. The results will then be discussed in the context of methodological, substantive and scientific reflections, followed bythe limitations of the study. The chapter concludes by discussing the possibilities for future research involving empirical studies specifically focused on sentiment analysis and/or machine learning, as well as suggestions for policy and practice, and future development work

7.2 Summary of the dissertation

Chapter 1 discussed the aim of the research as providing a comparative benchmark between four sentiment analysis approaches in a distributed cluster environment with eight machines. This research specifically focused on the social media aspect of Big Data analytics, and therefore made use of the Twitter streaming API as a data source of (what was expected to be) opinionated texts.

The problem statement was then posed later in Chapter 1, which presented the argument that sentiment analysis research generally focuses on classifier performance metrics (whether accuracy as a percentage, or F-measure as a combination of precision and recall) without considering the performance and resource usage aspects of a practical application. Such information is necessary for the planning and decision-making at the inception of a real-world sentiment analysis project, and the lack thereof means that every new implementation must also involve some form of experimentation to determine which approaches are suited to

specific environments - specifically in terms of aspects such as memory and processing requirements in addition to accuracy requirements.

The literature review in Chapter 2 investigated the current body of knowledge in terms of Big Data analytics as an overarching theme, and specifically examined the nature of social media research as a subset of Big Data. From there, sentiment analysis was investigated as a sub field of text classification as a means of analysing unstructured text data in social media data. From a literature search it was determined that the four most popular sentiment analysis approaches for research are Naïve-Bayes classifiers, lexicon-based classifiers, Neural Network classifiers and Support Vector Machine classifiers.

These classifiers were found to have been implemented and tested in a number of environments with a wide variety of data sources, but the majority of research focused on accuracy as the sole metric for comparison and improvement. This presented the analysis and comparison of resource requirements as a gap in current literature, especially so in distributed environments.

Based on the aim of the study and the type of data that was expected to be generated, the research methodology was formulated in Chapter 3 as experiments involving empirical analysis of quantitative metrics relating to real-world performance. This was followed by a discussion of the research process to be followed, as well as the procedure used for the analysis and interpretation of the data. The limitations of the study were then examined.

Chapter 4 discussed the research environment in which the experiments took place. This involved a cluster with a total of nine machines, of which eight were responsible for the computational aspect. The ninth machine was responsible for storage and some support services. The remainder of the chapter was devoted to the examination of the commodity software used on the machines to support the experiment, and the research software implementing the sentiment analysis approaches and interfacing with the distributed database.

A data pipeline was proposed in Chapter 5 for the purposes of the experiments, and each aspect of this pipeline was discussed in detail. From there an assessment of the expected work distribution as a function of the cluster size was documented, along with the expected impact that would have based on the cache sizes used by the software running the experiment. The

data storage aspect was then documented in terms of the database fields and tables, along with the overall database structure that was set up to store the data used for training and testing. A discussion of the measurements to be collected during training, validation and testing followed, and concluded with the limitations involved with the experimental design of the study.

The results of the study were then presented in Chapter 6 for each sentiment analysis approach individually, at a training, validation and testing level, across all cluster sizes. These results were then presented in a comparative format in terms of training and testing metrics as a series of sections, each involving a specific metric.

Based on the results obtained, the research questions originally posed in Chapter 1 can now be revisited.

7.2.1 RQ_1: Can bottlenecks in classifier performance be addressed by increasing the number of machines?

Of the four classifiers, only the SVM classifier experienced a CPU bottleneck. This is visible in the CPU time measurement for the classifier reflecting a number corresponding to the number of cores of each machine. In this case, adding more machines would add more cores, and therefore improve the overall throughput, but it would not address the bottleneck itself, i.e., the limiting factor will remain the CPU, and the other resources of each machine, such as the I/O would remain under-utilised. In such a case it would rather be recommended to scale the CPU performance of one machine vertically, such that it would match the speed at which the data source (in this case the database) could supply new data for classification. In such a situation the per-machine performance would be more effective. At that point the combined throughput could then be increased by adding more machines once the effect ceiling of vertical scalability had been reached.

The other three classifiers were all limited by I/O throughput. The CPU time in each case was always roughly one second, meaning that the processor was at approximately 25% capacity for the duration of the experiments. The reason for this is that the data source could not supply sufficient data to saturate all four cores on all the machines, regardless of the number of machines taking part in the experiment. There are two ways in which this could be addressed:

1) Increase the cache size of each processing thread. For all experiments, the cache size was set at five hundred Tweets per thread. Increasing this number would reduce the time spent creating and starting a thread, allowing each thread to spend more time classifying Tweets. This should be balanced based on the input size of the dataset under classification, and the number of cores on each machine. That is, if 10 000 Tweets need to be classified, it would not be optimal to have a 5 000 tweet cache per thread on a four core machine, as that would leave two cores idle. In the case of the three classifiers which are I/O bound, it is expected that a larger input size combined with increased cache capacity would increase the throughput of each classifier.
2) The database supplying the data to the classifiers could be offloaded to dedicated machines, and fitted with faster storage (such as solid-state drives) in order to improve the speed at which the database can provide the classifiers with data. Should such changes still be insufficient to saturate the classifiers with data, an in-memory database could be considered instead. It is expected that this solution would not be sufficient on its own, and that an increase in thread cache size would be required regardless, in the case of fast classifiers, as the time spent creating and destroying threads offsets the benefit of parallelism on each machine.

Increasing the number of machines would, therefore, in both cases increase the overall throughput, but would not offset the limitations imposed by the bottlenecks associated with each classifier. It would be ideal to rather address per-machine performance until each machine is utilised optimally prior to increasing the number of machines. Such an approach would result in the greatest performance gains per machine added, and would likely be the most cost-effective for horizontal scalability.

7.2.2 RQ_2: How does the number of machines used affect the data throughput per machine?

All four of the classifiers scaled well as the number of machines were increased, although there was a slight reduction in per-machine performance across the board. The expectation is that this is due to two factors:

1) The Cassandra database is distributed across all the machines that participate in the experiment. As the number of machines is increased, so does the coordination traffic between the Cassandra instances. It is also more likely then, that the machine that needs

to coordinate the traffic for a particular transaction is not the local machine, since the replication factor of three stays constant regardless of whether three or eight machines participate in the experiment.

2) The NFS server on the ninth machine provides the shared storage for the logging output of all the machines. It is possible that the combined log traffic on such a bottleneck could slow down the overall performance of the cluster slightly.

The second factor could possibly be limited by switching to a distributed filesystem as well, although that would exacerbate the problems associated with coordination traffic and synchronisation, similarly to the issues experienced with the distributed database. An alternative option would be to host the database, network storage and computation on different machines altogether. In such a case the network capacity itself would likely prove to be the bottleneck, in the case of very large clusters.

Separating only the database and computation aspects of the experiment would likely be more advantageous in terms of near-linear scalability, since the database transactions involved in this experiment cause many read and write operations on the same disks that are used to run the experiment software. Additionally, the Cassandra commit log and data directories resided on the same disks on any given machine, whereas a separation in this regard could considerably increase performance.

7.2.3 RQ_3: Is there an inverse correlation between the throughput and accuracy of the algorithms for each metric (i.e. does a faster algorithm have poorer accuracy)?

This was not generally found to be the case. First of all, the lexicon-based classifier used the least resources and performed classification faster than the other classifiers, and achieved the poorest accuracy, as previously expected. The other three classifiers did not, however, conform to this expectation. The Neural Network and Naïve-Bayes classifiers used similar amounts of resources, and performed at similar speeds, but the Neural Network classifier performed far better in terms of accuracy. The SVM classifier which used the considerably more resources (especially in terms of CPU time) did not exceed the accuracy of the Neural Network classifier.

The lower accuracy of the SVM classifier may be due to the fact that this study did not tune the hyperparameters of the classifier (or any of the three machine learning classifiers, for that matter). It could be expected that, with correct tuning, the accuracy of the SVM classifier could exceed that of the Neural Network classifier (given its own tuning as well), but the question then arises whether there is considerable room for improvement above what the Neural Network already achieved, and whether that difference is worth the increase in resources and time required to run the SVM classifier.

7.2.4 RQ_4: Is there an inverse correlation between the training time and training resource usage, and accuracy (i.e. does a less accurate algorithm train faster)?

The results for training time and resource usage during training mimic that of the testing outcomes. The Naïve-Bayesian classifier trains the fastest and uses the least amount of resources, while performing poorest in accuracy testing. The SVM classifier uses the most resources, and trains considerably slower than the other two, and comes second in accuracy. The Neural Network, on the other hand, provides the highest accuracy, and uses the second most resources and takes the second longest to train. Furthermore, the SVM and Neural Network classifiers were found to saturate a single CPU core during training, meaning that they are CPU-bound by single-core performance. The Naïve-Bayesian classifier did not saturate a single CPU core, and is therefore I/O-bound. The lexicon-based classifier was excluded from this test as it did not make use of machine learning techniques.

7.3 Discussion of results

This subsection will review the lessons learned and the outcomes of the research from three different viewpoints: methodologically (how the research approach influenced the results), substantively (how this research compares to other research in the field) and scientifically (how this research has contributed to the scientific body of knowledge).

7.3.1 Methodological reflection

The choice to not tune the hyperparameters for each machine learning approach has somewhat limited the usefulness of the classifier performance metrics to some extent, and may have also influenced some of the training performance metrics if applied. The fact that none of the

approaches received any tuning does ensure that, comparatively, the results remain useful, but are not canonical values that can be taken at face value at an individual approach level.

7.3.2 Substantive reflection

Some aspects that have come to light since the literature study for this research was completed include new developments in terms of deep learning, distributed systems such as Tensorflow, new databases such as ScyllaDB (Angell, 2020) - a clone of Apache Cassandra that has been rewritten in C for better performance - and the development of more accurate lexicon-based sentiment analysis techniques and enhancements. Better performance monitoring tools such as NetData (NetData, 2020) and Prometheus (Prometheus, 2020) have also been developed, making the measurement of the metrics involved considerably simpler.

The decision to make use of Twitter data, and setting up automatic labelling of the data using emoticons is common practice, but there is room for improvement, nonetheless. The very common use of Twitter data for Big Data sentiment analysis research may be detrimental to the research field in the long run, since few studies make use of other data sources. This focus on Twitter data specifically may lead to a long-term bias in results (whether pertaining to accuracy or other measures) towards the kind of language and restrictions that microblogging imposes. While the use of Twitter data for this study makes the results more widely applicable to other researchers, who have likely also encountered Twitter data in their own environment, the impact that such a narrow focus may have on the field will likely become evident with time. The overall lack of data variety in this kind of research is likely to be exacerbated by the introduction of stricter legislation around the use of individuals' data (such as the European General Data Protection Regulation or the South African Protection of Personal Information Act) limiting the access and permissions that researchers have to social data.

7.3.2.1 RO_1: Determine the gap in current literature

In terms of research objectives, this study successfully completed RO_1: Determine the gap in current literature. This gap informed the direction that this study needed to take, and also informs future research that may be useful in addressing the current gap beyond the work completed in this study.

7.3.3 Scientific reflection

The results of this study demonstrated the value of empirical testing for sentiment analysis running on distributed systems. A number of notable conclusions could be drawn, largely specific to the environment of this study, such as the identification of bottlenecks and the impact of cluster size on them, the changes in throughput as the number of machines are increased and the investigation of a relationship between performance and accuracy of a range of classifiers.

It was found that, in most cases bottlenecks could be alleviated by adding more machines, but that the bottlenecks themselves remained as more machines were added, hampering the performance of the cluster as a whole, and that optimal performance could only be gained by scaling the resources per-machine in such a way that all of the resources would be utilised equally prior to scaling horizontally. This may in some cases be difficult to apply practically, since the bottlenecks can vary between different techniques or classifiers.

Furthermore, it was found that the various classifiers scaled well as the number of machines were increased, since they ran mostly independently of one another, with the NFS server and the Cassandra database cluster being the only shared resources between them. Removing the Cassandra database nodes from the machines participating in the experiment would likely increase the performance of both the classifiers and the database, since they currently compete for resources on the same machines.

No clear relationship between classifier accuracy and performance could be found, with a mixture of results between the classifiers. The only classifier that adhered well to the expectation that a faster classifier would provide poorer accuracy was the lexicon-based AFINN classifier, which used considerably less resources than the other three classifiers, which were all based on machine learning techniques.

Two research objectives are also applicable at this stage, RO_2 and RO_3.

7.3.3.1 RO_2: Compare the resource usage of four sentiment analysis approaches

This study successfully compared the resources usage of four sentiment analysis approaches during training, validation and testing, across a number of metrics. The measurement and completion of this comparison fulfils this research objective.

7.3.3.2 RO_3: Compare the classifier performance of four sentiment analysis approaches

This study successfully measured and reported on a variety of classifier performance metrics for all four sentiment analysis approaches, and made use of several statistical tests to confirm the hypotheses stated at the inception of the experiment. Based on the successful completion of these aspects of the study the research objective can be considered fulfilled. With this, all three research objectives originally stated in Chapter 1 have been fulfilled.

7.4 Limitations

This study was limited to off-the-shelf commodity desktop hardware. It is likely that some differences would result from the use of server hardware, virtualisation or cloud environments – all of which are used to some degree in commercial and HPC environments. Furthermore, the limited number of machines means that major extrapolation from the results of this study is likely to be inaccurate, rendering it less useful for large environments with hundreds of machines.

Performance can be highly dependent on the programming language used to develop a classifier. The use of Python for all classifiers allowed the use of many existing libraries widely used in scientific and commercial settings, and therefore provided a relatable starting point for comparison, but it also limits the conclusions drawn from this study to Python classifiers.

This study was limited to the four most popular sentiment analysis approaches in literature. Further investigation into the performance of other approaches is warranted to supplement the results of this study and broaden the comparative value of such research. The decision not to tune the hyperparameters of the classifiers, while deliberate for this study, does limit the comparative value of specifically the classifier performance metrics obtained, as the tuning of these parameters could have considerable implications for accuracy. The results of this study

remain indicative of performance and its relationship to accuracy, but the possible relationship between the two merits further investigation.

The use of one distributed database allowed for comparative measurements between classifiers, but the addition of multiple database backends would expand the understanding of the impact that the choice of database has on the end result. An investigation into the addition of an in-memory database would likely be especially useful in identifying a possible I/O bottleneck caused by slower storage.

The number of metrics included in the performance comparison aspect of this study was largely limited based on tools available for measurement. The addition of metrics for factors such as power consumption and heat generation may prove valuable for large scale deployments, and could be used to calculate cost estimations for sentiment analysis approaches in distributed environments.

This study focused specifically on Twitter data as a manifestation of a form of Big Data in a social media context, which limited the classifications to 140 characters at the document-level. Twitter has increased the maximum number of characters per Tweet to 280 characters since this study has collected data, but that is still insufficient to determine whether the results of this study also apply to larger documents, such as Facebook posts or forum discussions. The effects of expanding the classification applied in this study to include aspect or sentence level classification on the results is also unknown.

7.5 Recommendations and future work

Based on the research and implementation process, and based on the results obtained, it is possible to propose some recommendations in terms of policy and practice, future research and future development work. This section will discuss these ideas in detail, and explain the reasoning in favour of such recommendations.

7.5.1 Policy and practice

The use of Twitter as a data source for social media Big Data for this study largely stemmed from a number of factors relating to access and permissions. Twitter allows easy (and free of

charge) access to the streaming API, which can provide a stream of up to 1% of all Tweets posted globally, which is sufficient to gather a large amount of data in a short period of time as long as the filtering on the incoming data is set up in such a way that it does not needlessly limit the type of data requested. Furthermore, Twitter also grants nearly blanket permission to use Twitter data for research purposes, and the explicit consent therefore is obtained from every user at the point of registration (as part of the standard terms and conditions and privacy policy) on the platform, relieving a large burden from the researcher's side.

Other major platforms such as Facebook and WhatsApp are not nearly as accessible to researchers (likely due to the more private nature of these services), and it may lead to biases in research outcomes if sentiment analysis research on social media data generally focuses solely on Twitter data due to its convenience compared to other platforms. The alternative to an open API is to crawl platforms for data, which may be in contravention of terms of service and is generally cumbersome to do. Such activities may also be subject to throttling, which would further hamper data collection. The collection of data in such a manner may also be in contravention of laws surrounding the use of personal data without explicit consent, with new legislation rolling out all over the world in favour of the individual whose data is being stored and processed.

Alternatively, researchers may opt to investigate other data sources, such as wiki services (where content is often made available on an open license) and open access publications (often made available through APIs such as OAI-PMH), although neither of these options are generally as opinionated (and therefore useful for sentiment analysis) as social media.

An investigation into the possible social media research data sources and the legal and ethical restrictions surrounding them (especially considering recent legislation) would be invaluable in expanding the applications of sentiment analysis (and to an extent Big Data analytics) research.

7.5.2 Further research

This study was limited to four techniques, one database and nine machines (of which eight were computationally involved). Future research could expand on this study by investigating other techniques, tuning the hyperparameters for a technique and exploring the impacts they

have on the results, and expanding the number of machines participating the in experiment. The resource specifications of the individual machines could also be changed to understand how improvements in e.g. I/O performance impacts processor saturation, or testing what the impact is of increased network bandwidth, or how an increase in latency affects the outcome.

This study also focused solely on Twitter data, which means that the impact of longer form text documents such as comprehensive movie reviews or book chapters on this process is still unknown. Longer texts also allow for testing classification at aspect or sentence level. The use of machine learning for classification also applies to other fields such as image recognition, where an empirical study of this nature may be equally valuable.

Future studies could consider the addition of metrics such as power consumption and heat generation to the existing set of metrics measured by this study. Such information could drive decision-making for deployments making use of more machines based on cost estimations and environmental factors.

This study limited each set of test runs to ten runs at every cluster size, resulting in a total of 240 test runs. An increase in the number of test runs would likely average out the variance between the runs observed and may provide a more accurate real-world picture of what could be expected in the field.

7.5.3 Further development work

Classifiers developed in other programming languages are likely to perform differently, and it would therefore be worth investigating the real impact that the choice of language has on a distributed classifier. Such an investigation could also be extended to include new frameworks such as Tensorflow (Abhadi, Barham, Chen, et al., 2016), which handle data/work distribution on behalf of the developer, or the implementation of entirely new libraries and frameworks if appropriate.

7.6 Summary

This chapter presented a summary of the study as a whole, and then proceeded to evaluate the research questions posed in Chapter 1. This was following by the methodological, substantive

and scientific reflections on the work completed and the results obtained. The chapter was concluded with a discussion of the limitations of this study and recommendations for policy and practice, future research, and future development work. This study demonstrated the complete process of setting up a distributed NoSQL environment for the purpose of comparatively testing four sentiment analysis techniques in a range of cluster sizes. The results of over two hundred experimental runs were then summarised and discussed at length, and addressed the research questions posed at the inception. Future studies should be undertaken to investigate the impact that changes in the research environment, programming language and parameter tuning has on sentiment analysis processes in distributed environments.

REFERENCE LIST

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265-283).

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). *Sentiment analysis of Twitter data*. 30–38. Retrieved from http://dl.acm.org/citation.cfm?id=2021109.2021114

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*(1), 37–66. https://doi.org/10.1007/BF00153759

Alam, M., Muley, A., Kadaru, C., & Joshi, A. (2013). *Oracle NoSQL Database: Real-Time Big Data Management for the Enterprise*. New York, New York, USA: McGraw Hill Professional.

Angell, J. (2020). *Scylladb | the real-time big data database*. ScyllaDB. https://www.scylladb.com/

Apache Software Foundation. (2020). Apache Cassandra. http://cassandra.apache.org/

Asur, S., & Huberman, B. A. (2010). *Predicting the Future with Social Media*. *1*: 492–499. https://doi.org/10.1109/WI-IAT.2010.63

Bagga, S., & Sharma, A. (2018). Big Data and Its Challenges: A Review. *4th International Conference on Computing Sciences (ICCS)*, 183–187. https://doi.org/10.1109/ICCS.2018.00037

Bartz-Beielstein, T., Chiarandini, M., Paquete, L., & Preuss, M. (2010). *Experimental Methods for the Analysis of Optimization Algorithms* (Vol. 2). Retrieved from http://link.springer.com/content/pdf/10.1007/978-3-642-02538-9.pdf

Berry, M. (2004). *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer New York. https://doi.org/10.1007/978-1-84800-046-9

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc.

Bouazizi, M., & Ohtsuki, T. (2016). Sentiment analysis: From binary to multi-class classification: A pattern-based approach for multi-class sentiment analysis in Twitter. *2016 IEEE International Conference on Communications (ICC)*, 1–6. https://doi.org/10.1109/ICC.2016.7511392

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771.

https://doi.org/10.1016/J.PATCOG.2004.03.009

Boyd, D., & Crawford, K. (2012, June). Critical Questions for Big Data. *Information, Communication & Society*, 15, 662–679. https://doi.org/10.1080/1369118X.2012.678878

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167. https://doi.org/10.1023/A:1009715923555

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, *39*(4), 12. https://doi.org/10.1145/1978915.1978919

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3), 1-27.

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data To Big Impact. *Mis Quarterly*, *36*(4), 1165–1188.

Cochran, W. G. (1950). The comparison of percentages in matched samples. *Biometrika*. https://doi.org/10.1093/biomet/37.3-4.256

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed). Retrieved from http://www.amazon.com/Introduction-Algorithms-Edition-Thomas-Cormen/dp/0262033844

Datastax. (2018a). *About Apache Cassandra*. Retrieved October 14, 2018, from https://docs.datastax.com/en/cassandra/3.0/cassandra/cassandraAbout.html

Datastax. (2018b). *Architecture in brief*. Retrieved October 14, 2018, from https://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archIntro.html

Davenport, T. H. (2006). Competing on Analytics. *Harvard Business Review*, *84*(1), 98–107.

Dede, E., Sendir, B., Kuzlu, P., Hartog, J., & Govindaraju, M. (2013). An evaluation of Cassandra for Hadoop. *IEEE International Conference on Cloud Computing, CLOUD*, 494–501. https://doi.org/10.1109/CLOUD.2013.31

Del Vecchio, P., Di Minin, A., Petruzzelli, A. M., Panniello, U., & Pirri, S. (2018). Big data for open innovation in SMEs and large corporations: Trends, opportunities, and challenges. *Creativity and Innovation Management*, *27*(1), 6–22. https://doi.org/10.1111/caim.12224

Donovan, A. A. A., & Kernighan, B. W. (2015). *The Go Programming Language* (1st ed.). Addison-Wesley Professional.

Duch, W., & Jankowski, N. (1997). New neural transfer functions. *Applied Mathematics and*

Computer Science, *7*, 639–658. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.5132&rep=rep1&type=pdf

Duncan, B., & Zhang, Y. (2015). Neural networks for sentiment analysis on Twitter. *IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, 275–278. https://doi.org/10.1109/ICCI-CC.2015.7259397

Elgendy, N., & Elragal, A. (2014). *Big Data Analytics: A Literature Review Paper*. https://doi.org/10.1007/978-3-319-08976-8_16

Elminaam, D. S. A., Abdual-Kader, H. M., & Hadhoud, M. M. (2010). Evaluating The Performance of Symmetric Encryption Algorithms. *IJ Network Security*, *10*(3), 216–222.

Esuli, A., & Sebastiani, F. (2006, May). Sentiwordnet: A publicly available lexical resource for opinion mining. *LREC* (Vol. 6, pp. 417-422).

Facebook. (2020). Facebook. https://www.facebook.com/

Fan, W., & Gordon, M. D. (2014). The power of social media analytics. *Communications of the ACM*, *57*(6), 74–81. https://doi.org/10.1145/2602574

Fernández-Gavilanes, M., Álvarez-López, T., Juncal-Martínez, J., Costa-Montenegro, E., & Javier González-Castaño, F. (2016). Unsupervised method for sentiment analysis in online texts. *Expert Systems with Applications*, *58*, 57–75. https://doi.org/10.1016/j.eswa.2016.03.031

Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, *73*(2), 133–153. https://doi.org/10.1007/s10994-008-5064-8

Gamallo, P., & Garcia, M. (2014). Citius: A Naïve-Bayes Strategy for Sentiment Analysis on English Tweets. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 171–175. https://doi.org/10.3115/v1/S14-2026

Gessert, F., Wingerath, W., Friedrich, S., & Ritter, N. (2017). NoSQL database systems: a survey and decision guidance. *Computer Science - Research and Development*, *32*(3–4), 353–365. https://doi.org/10.1007/s00450-016-0334-3

Ghani, N. A., Hamid, S., Targio Hashem, I. A., & Ahmed, E. (2019). Social media Big Data analytics: A survey. *Computers in Human Behavior*, *101*, 417–428. https://doi.org/10.1016/J.CHB.2018.08.039

Ghiassi, M., Skinner, J., & Zimbra, D. (2013). Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial Neural Network. *Expert Systems*

with Applications, *40*(16), 6266–6282. https://doi.org/10.1016/j.eswa.2013.05.057

Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1–12.

Go, A., Huang, L., Bhayani, R., & Twitter. (2009). Twitter Sentiment Analysis. In *Entropy* (Vol. 2009). https://doi.org/10.1007/978-3-642-35176-1_32

Guo, Y., Biczak, M., Varbanescu, A., & Iosup, A. (2013). *How well do graph-processing platforms perform? an empirical performance evaluation and analysis*. Retrieved from http://www.pds.ewi.tudelft.nl/~iosup/perf-eval-graph-proc14ipdps.pdf

Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S., & Saxena, U. (2017). NoSQL databases: Critical analysis and comparison. *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, 293–299. https://doi.org/10.1109/IC3TSN.2017.8284494

Ha, I., Back, B., & Ahn, B. (2015). MapReduce Functions to Analyze Sentiment Information from Social Big Data. *International Journal of Distributed Sensor Networks*, *11*(6), 417-502. https://doi.org/10.1155/2015/417502

Hatzivassiloglou, V., & McKeown, K. R. (1997). *Predicting the semantic orientation of adjectives*. https://doi.org/10.3115/979617.979640

Haykin, S. (2004). A comprehensive foundation. *Neural Networks*, *2*(2004), 41.

He, Y., & Zhou, D. (2011). Self-training from labeled features for sentiment analysis. *Information Processing & Management*, *47*(4), 606–616. https://doi.org/10.1016/J.IPM.2010.11.003

Hu, X., Tang, J., Gao, H., & Liu, H. (2013). Unsupervised sentiment analysis with emotional signals. *Proceedings of the 22nd International Conference on World Wide Web - WWW '13*, 607–618. https://doi.org/10.1145/2488388.2488442

Huang, S., Peng, W., Li, J., & Lee, D. (2013). Sentiment and topic analysis on social media. *Proceedings of the 5th Annual ACM Web Science Conference on - WebSci '13*, 172–181. https://doi.org/10.1145/2464464.2464512

Instagram. (2020). Instagram. https://instagram.com/

Joachims, T. (1998). Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. *Proceedings of the 10th European Conference on Machine Learning ECML '98*. https://doi.org/10.1007/BFb0026683

Khan, M. A., Uddin, M. F., & Gupta, N. (2014). Seven V's of Big Data understanding Big Data to extract value. *Proceedings of the 2014 Zone 1 Conference of the American*

Society for Engineering Education, 1–5. https://doi.org/10.1109/ASEEZone1.2014.6820689

Kharde, V. A., & Sonawane, P. S. (2016). *Sentiment Analysis of Twitter Data: A Survey of Techniques*. https://doi.org/10.5120/ijca2016908625

Khuc, V. N., Shivade, C., Ramnath, R., & Ramanathan, J. (2012). Towards Building Large-scale Distributed Systems for Twitter Sentiment Analysis. *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 459–464. https://doi.org/10.1145/2245276.2245364

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007). Ensembles of Multi-Objective Decision Trees. In *Machine Learning: ECML 2007* (pp. 624–631). https://doi.org/10.1007/978-3-540-74958-5_61

Kouloumpis, E., Wilson, T., & Moore, J. (2011). Twitter sentiment analysis: The good the bad and the omg! *ICWSM*, *11*, 538–541.

Krishnan, K. (2013). *Data Warehousing in the Age of Big Data* (1st ed). Amsterdam: Elsevier, Morgan Kaufmann.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., … Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. *International Conference on Artificial Neural Networks*, 53–60. Retrieved from http://mleg.cse.sc.edu/edu/csce822/uploads/Main.ReadingList/KNN_recognition.pdf

Lexalytics. (2015). *Sentiment Analysis* | lexalytics. Retrieved April 29, 2015, from http://www.lexalytics.com/technical-info/sentiment-analysis

Lin, K. H.-Y., Yang, C., & Chen, H.-H. (2007). What emotions do news articles trigger in their readers? *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '07*, 733. https://doi.org/10.1145/1277741.1277882

Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, *5*(1), 1–167. https://doi.org/10.2200/S00416ED1V01Y201204HLT016

Liu, B., Blasch, E., Chen, Y., Shen, D., & Chen, G. (2013). Scalable sentiment classification for Big Data analysis using Naïve Bayes Classifier. *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*, 99–104. https://doi.org/10.1109/BigData.2013.6691740

Liu, S. M., & Chen, J.-H. (2015). A multi-label classification based approach for sentiment

classification. *Expert Systems with Applications*, *42*(3), 1083–1093. https://doi.org/10.1016/J.ESWA.2014.08.036

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16–32. https://doi.org/10.1016/J.COSREV.2017.10.002

Martín-Valdivia, M.-T., Martínez-Cámara, E., Perea-Ortega, J.-M., & Ureña-López, L. A. (2013). Sentiment polarity detection in Spanish reviews combining supervised and unsupervised approaches. *Expert Systems with Applications*, *40*(10), 3934–3942. https://doi.org/10.1016/J.ESWA.2012.12.084

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*. https://doi.org/10.1007/BF02295996

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Miller, G. A. (1995). *WordNet: A lexical database for English. Association for Computing Machinery*. https://doi.org/10.1145/219717.219748

Minelli, M., Chambers, M., & Dhiraj, A. (2013). *Big Data, Big Analytics* (1st ed). New Jersey: Wiley.

Mohammad, S., Svetlana, K., & Zhu, X. (2013). Sentiment Analysis of Tweets. Retrieved March 16, 2014, from *Proceedings of the seventh international workshop on Semantic Evaluation Exercises* website: http://www.umiacs.umd.edu/~saif/WebPages/Abstracts/NRC-SentimentAnalysis.htm

Mohey, D., & Hussein, E. M. (2016). A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences*, (April). https://doi.org/10.1016/j.jksues.2016.04.002

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2013). *Foundations of Machine Learning*. Cambridge, Massachusetts: The MIT Press.

Moniruzzaman, A. B. M., & Hossain, S. A. (2013). NoSQL Database : New Era of Databases for Big data Analytics- Classification, Characteristics and Comparison. *International Journal of Database Theory and Application*, pp. 1–13.

Mouton, J. (2001). *How to succeed in your master's and doctoral studies : a South African guide and resource book*. Pretoria: Van Schaik.

Narr, S., Hulfenhaus, M., & Albayrak, S. (2012). Language-Independent Twitter Sentiment

Analysis. *Proceedings of KDML-2012, the 2012 Workshop on Knowledge Discovery, Data Mining and Machine Learning*.

Neethu, M. S., & Rajasree, R. (2013). Sentiment analysis in Twitter using machine learning techniques. *Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 1–5. https://doi.org/10.1109/ICCCNT.2013.6726818

Netdata. (2020). *Netdata—Get control of your linux servers. Simple. Effective. Awesome.* Netdata. https://www.netdata.cloud

Nielsen, F. Å. (2011). A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, 93-98.

Nodarakis, N., Tsakalidis, A., Sioutas, S., & Tzimas, G. (2016). Large scale sentiment analysis on Twitter with Spark. *1st International Workshop on Multi-Engine Data Analytics, 1-8.*

Oates, B. J. (2005). *Researching Information Systems and Computing*. Retrieved from https://books.google.co.za/books?id=VyYmkaTtRKcC

Ortigosa-Hernández, J., Rodríguez, J. D., Alzate, L., Lucania, M., Inza, I., & Lozano, J. A. (2012). Approaching Sentiment Analysis by using semi-supervised learning of multi-dimensional classifiers. *Neurocomputing*, *92*, 98–115. https://doi.org/10.1016/j.neucom.2012.01.030

Pak, A., & Paroubek, P. (2010). *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*. Retrieved from https://www.aclweb.org/anthology/L10-1263/

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends® in Information Retrieval*, *2*(1–2), 1–135. https://doi.org/10.1561/1500000011

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 79-86.

Papaefthymiou, M., & Rodrigue, J. (1994). Implementing Parallel Shortest-Path Algorithms. *Parallel Algorithms: Third DIMACS Implementation Challenge, 30,* 59-68.

Pedregosa, F., Varoquaux, G. G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Potlapally, N. R., Ravi, S., Raghunathan, A., & Jha, N. K. (2006). A study of the energy

consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing*, *5*(2), 128–143.

Pozzi, F. A., Fersini, E., Messina, E., & Liu, B. (2016). Sentiment Analysis in Social Networks. Morgan Kaufmann.

Prometheus. (2020). *Prometheus—Monitoring system & time series database*. https://prometheus.io/

Puppetlabs. (2020). Powerful infrastructure automation and delivery | Puppet. https://puppet.com/

Raschka, S. (2015). *Python Machine Learning*. https://doi.org/10.1007/s13398-014-0173-7.2

Raschka, S. (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*, *3*(24), 638. https://doi.org/10.21105/joss.00638

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359. https://doi.org/10.1007/s10994-011-5256-5

Roebuck, K. (2012). *Sentiment Analysis: High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Retrieved from https://books.google.com/books?id=kqsNBwAAQBAJ&pgis=1

SAS. (2015). S*AS Sentiment Analysis* Retrieved April 29, 2015, from http://www.sas.com/en_us/software/analytics/sentiment-analysis.html

Scopus. (2020). *Elsevier Scopus*. https://www.scopus.com/

Sedgewick, R. (1998). *Algorithms in C++, Parts 1-4: Fundamentals, Data Structure, Sorting, Searching*. Retrieved from https://books.google.com/books?id=ZCchAeprwvYC&pgis=1

Sentimeter. (2015). *SentiMeter | Welcome.* Retrieved April 29, 2015, from https://sentimeter.com/business/welcome

Sharma, A., & Dey, S. (2012). A document-level sentiment analysis approach using artificial Neural Network and sentiment lexicons. *ACM SIGAPP Applied Computing Review*, *12*(4), 67–75. https://doi.org/10.1145/2432546.2432552

Sindhwani, V., & Melville, P. (2008). Document-Word Co-regularization for Semi-supervised Sentiment Analysis. *Eighth IEEE International Conference on Data Mining*, 1025–1030. https://doi.org/10.1109/ICDM.2008.113

Snapchat. (2020). https://www.snapchat.com

Solid IT. (2019). *DB-Engines Ranking - popularity ranking of wide column stores*. Retrieved November 24, 2019, from https://db-engines.com/en/ranking/wide+column+store

Stewart, R., & Singer, J. (2012). *Comparing fork/join and MapReduce*. 1–20. Retrieved from http://www.macs.hw.ac.uk/cs/techreps/docs/files/HW-MACS-TR-0096.pdf

Taboada, M. (2016). Sentiment Analysis: An Overview from Linguistics. *Annual Review of Linguistics*, *2*(1), 325–347. https://doi.org/10.1146/annurev-linguistics-011415-040518

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, *37*(2), 267–307. https://doi.org/10.1162/COLI_a_00049

The Stanford NLP Group. (2015). *The Stanford NLP (Natural Language Processing) Group.* Retrieved April 29, 2015, from http://nlp.stanford.edu/

Tripathy, A., Agrawal, A., & Rath, S. K. (2016). Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, *57*, 117–126. https://doi.org/10.1016/J.ESWA.2016.03.028

Troussas, C., Virvou, M., Espinosa, K. J., Llaguno, K., & Caro, J. (2013). Sentiment analysis of Facebook statuses using Naïve Bayes classifier for language learning. *IISA 2013*, 1–6. https://doi.org/10.1109/IISA.2013.6623713

Tsai, C.-W., Lai, C.-F., Chao, H.-C., Vasilakos, A. V. (2015). Big data analytics: a survey. *Journal of Big Data*, *2*(1), 21. https://doi.org/10.1186/s40537-015-0030-3

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 30–44. Retrieved from http://lpis.csd.auth.gr/publications/tsoumakas-mmd08.pdf

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2011). Random k-Labelsets for Multilabel Classification. *IEEE Transactions on Knowledge and Data Engineering*, *23*(7), 1079–1089. https://doi.org/10.1109/TKDE.2010.164

Twitter Inc. (2013). Documentation | Twitter Developers. Retrieved March 16, 2014, from https://dev.Twitter.com/docs

Twitter. (2020a). Twitter. https://twitter.com/

Twitter. (2020b). Twitter—Privacy policy. Twitter - Privacy Policy. https://twitter.com/content/twitter-com/legal/en/privacy.html

Van der Linde, I., & Kotzé, E. (2018). Design and evaluation of an artefact for realtime Twitter sentiment analysis : lessons Learnt. *Journal for New Generation Sciences*, *16*(1),

129–144. Retrieved from https://journals.co.za/content/journal/10520/EJC-151bd3ce3e

Vapnik, V. N. (2000). *The nature of statistical learning theory*. Retrieved from https://books.google.co.za/books/about/The_Nature_of_Statistical_Learning_Theor.html?id=sna9BaxVbj8C&redir_esc=y&hl=en

Varghese, R., & Jayasree, M. (2013). Aspect based Sentiment Analysis using support vector machine classifier. *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1581–1586. https://doi.org/10.1109/ICACCI.2013.6637416

Wang, H., Can, D., Kazemzadeh, A., Bar, F., & Narayanan, S. (2012). *A system for real-time Twitter sentiment analysis of 2012 U.S. presidential election cycle*. 115–120. Retrieved from http://dl.acm.org/citation.cfm?id=2390470.2390490

Xia, R., Zong, C., & Li, S. (2011). Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, *181*(6), 1138–1152. https://doi.org/10.1016/J.INS.2010.11.023

Yvonne Feilzer, M. (2010). Doing Mixed Methods Research Pragmatically: Implications for the Rediscovery of Pragmatism as a Research Paradigm. *Journal of Mixed Methods Research*, *4*(1), 6–16. https://doi.org/10.1177/1558689809349691

Zhang, L., Ghosh, R., Dekhil, M., Hsu, M., & Liu, B. (2011). Combining lexicon-based and learning-based methods for Twitter sentiment analysis. *HP Laboratories, Technical*. Retrieved from http://www.hpl.hp.com/techreports/2011/HPL-2011-89.pdf

Zhang, M.-L., & Zhou, Z.-H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048. https://doi.org/10.1016/J.PATCOG.2006.12.019

Zikmund, W., Babin, B., Carr, J., & Griffin, M. (2010). Business Research Methods 8th edition. Nelson Education.

APPENDICES

Appendix A: CQL listing for Cassandra database table schemas

```
CREATE KEYSPACE Twitter WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '3'}  AND
durable_writes = true;

CREATE TABLE Twitter.train4 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train5 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
```

```
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train6 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train7 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
```

```
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train0 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train1 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
```

```
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train2 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train3 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
```

```
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.test (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    rownum bigint,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train8 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
```

```
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.train9 (
    uuid timeuuid PRIMARY KEY,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE Twitter.output (
    uuid timeuuid PRIMARY KEY,
    classification boolean,
    datetime timestamp,
    tweetauthor text,
    tweetid bigint,
    tweettext text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
```

```
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

Appendix B: Ethical clearance certificate

UNIVERSITY OF THE FREE STATE
UNIVERSITEIT VAN DIE VRYSTAAT
YUNIVESITHI YA FREISTATA

UFS·UV
NATURAL AND AGRICULTURAL SCIENCES
NATUUR- EN LANDBOUWETENSKAPPE

Faculty of Natural and Agricultural Sciences

11-Nov-2015

Dear **Mr Ian Van Der Linde**

Ethics Clearance: **A comparison of sentiment analysis techniques in a parallel and distributed NoSQL environment**

Principal Investigator: **Mr Ian Van Der Linde**

Department: **Computer Science and Informatics (Bloemfontein Campus)**

APPLICATION APPROVED

This letter confirms that a research proposal with tracking number: **UFS-HSD2015/0606** and title: **'A comparison of sentiment analysis techniques in a parallel and distributed NoSQL environment'** was given ethical clearance by the Ethics Committee.

Your ethical clearance number, to be used in all correspondence is: **UFS-HSD2015/0606**

Please ensure that the Ethics Committee is notified should any substantive change(s) be made, for whatever reason, during the research process. This includes changes in investigators. Please also ensure that a brief report is submitted to the Ethics Committee on completion of the research.
The purpose of this report is to indicate whether or not the research was conducted successfully, if any aspects could not be completed, or if any problems arose that the Ethics Committee should be aware of.

Note:

1. This clearance is valid from the date on this letter to the time of completion of data collection.
2. Progress reports should be submitted annually unless otherwise specified.

Yours Sincerely

Prof. PD (Danie) Vermeulen
Chairperson: Ethics Committee
Faculty of Natural and Agricultural Sciences

Natural and Agricultural Sciences Research Ethics Committee
Office of the Dean: Natural and Agricultural Sciences
T: +27 (0)51 401 2322 | F: +27 (0)51 401 3728 | E: heidemannj@ufs.ac.za
Biology Building, Ground Floor, Room 9 | P.O. Box/Posbus 339 (Internal Post Box G44) | Bloemfontein 9300 | South Africa
www.ufs.ac.za