Cape Peninsula University of Technology

REAL-TIME PROBABILISTIC REASONING SYSTEM USING LAMBDA ARCHITECTURE

by

ARINZE ANIKWUE

Thesis submitted in fulfilment of the requirements for the degree

Master of Technology: Information Technology

in the Faculty of Informatics and Design

at the Cape Peninsula University of Technology

Supervisor: Dr. Boniface Kabaso

District Six
June 2019

DECLARATION

I, Arinze Anikwue, declare that the contents of this dissertation/thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

[illegible]	March 19, 2020.
Signed	**Date**

ABSTRACT

The proliferation of data from sources like social media, and sensor devices has become overwhelming for traditional data storage and analysis technologies to handle. This has prompted a radical improvement in data management techniques, tools and technologies to meet the increasing demand for effective collection, storage and curation of large data set. Most of the technologies are open-source.

Big data is usually described as very large dataset. However, a major feature of big data is its velocity. Data flow in as continuous stream and require to be actioned in real-time to enable meaningful, relevant value. Although there is an explosion of technologies to handle big data, they are usually targeted at processing large dataset (historic) and real-time big data independently. Thus, the need for a unified framework to handle high volume dataset and real-time big data. This resulted in the development of models such as the Lambda architecture.

Effective decision-making requires processing of historic data as well as real-time data. Some decision-making involves complex processes, depending on the likelihood of events. To handle uncertainty, probabilistic systems were designed. Probabilistic systems use probabilistic models developed with probability theories such as hidden Markov models with inference algorithms to process data and produce probabilistic scores. However, development of these models requires extensive knowledge of statistics and machine learning, making it an uphill task to model real-life circumstances. A new research area called probabilistic programming has been introduced to alleviate this bottleneck.

This research proposes the combination of modern open-source big data technologies with probabilistic programming and Lambda architecture on easy-to-get hardware to develop a highly fault-tolerant, and scalable processing tool to process both historic and real-time big data in real-time; a common solution. This system will empower decision makers with the capacity to make better informed resolutions especially in the face of uncertainty.

The outcome of this research will be a technology product, built and assessed using experimental evaluation methods. This research will utilize the Design Science Research (DSR) methodology as it describes guidelines for the effective and rigorous construction and evaluation of an artefact.

Probabilistic programming in the big data domain is still at its infancy, however, the developed artefact demonstrated an important potential of probabilistic programming combined with Lambda architecture in the processing of big data.

Keywords: Big Data, big data processing, probabilistic reasoning, probabilistic programming, Lambda architecture.

ACKNOWLEDGEMENTS

I wish to thank:

- Yahweh, the source of all.
- Mr. Isaac Anikwue, my dad & Mrs. Miriam Anikwue, my mum for everything.
- My siblings for their continuous help, encouragement and love.
- Dr. Boniface Kabaso for his guidance and support.
- Prof. Israel & Ada Jideani for their encouragement and help.
- Colleagues and friends for their insight and support.
- The department of Information Technology, Cape Peninsula University of Technology for the platform and support.

DEDICATION

To my parents, Mr. Isaac and Mrs Miriam Anikwue, and to my siblings.

PUBLICATIONS FROM THIS RESEARCH

- Anikwue, A. & Kabaso, B. 2019. Probabilistic Programming and Big Data. In 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD). IEEE.

- Anikwue, A. & Kabaso, B. 2018. A systematic review of Lambda Architecture based big data solutions. In 2018 Conference on Information Communications Technology and Society (ICTAS).

GLOSSARY

Terms/Acronyms/Abbreviations	Definition/Explanation
DSR	Design Science Research
DSRM	Design Science Research Methodology
EPL	English Premiership League
FSS	Final Search String
GFS	Google File System
HDFS	Hadoop Distributed File System
IC	Inclusion/exclusion Criteria
OBJ	Objective
QAC	Quality Assessment Criteria
RQ	Research Question
SLR	Systematic Literature Review
SLRQ	Systematic Literature Review Question
VMP	Variational Message Research

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE
INTRODUCTION

1.1 Introduction

This chapter starts with a description of the research background and motivation for this study in Section 1.2, and then the research problem is listed in Section 1.3. Section 1.5 presents the research questions. The aim and objectives of this research are outlined in Section 1.5 and Section 1.6 discusses delineation and assumptions of this study. The proposed methodology for this research is described in Section 1.7. Section 1.8 ends this chapter with a descriptive structure of the thesis.

1.2 Background and Motivation

Data is a collection of facts or statistics in an unorganized form usually for calculation, analysis and/or planning. Data is limitless and ubiquitous. As McAfee & Brynjolfsson (2012) pointed out, individuals, businesses, institutions and organizations are accumulating and producing massive amount of data than they know what to do with as a by-product of business processes, website tracking, finance, accounting among others. This increasing amount of data generated on a daily basis usually originates from multiple sources like sensors and mobile devices, and in different formats.

Looking at data generated online, Fan & Bifet (2013, p. 1) wrote: “The web pages indexed by Google were around one million in 1998, but quickly reached one billion in 2000 and have already exceeded one trillion in 2008.” This can be credited to social media applications like YouTube, Twitter, Instagram, etc., that allow its users to generate huge amount of data. Most of these data are continuously generated as streams and are volatile. This flood of data is called *Big Data* and according to Hansen (2013), it represents a significant innovation in data management. Big data simple put is data that is excessively large, very fast and thus, tough for extant traditional data management tools to process (Madden, 2012).

Laney (2001) indicated high volume, velocity and variety, popularly called the *3Vs*, as the three attributes that describe big data. This has formed the basis for most interpretations of big data. The cynosure of big data both in academia and industry has been on volume, albeit the significance of other Vs are recognised by many (Mishne et al., 2013).

Volume refers to the increasing size of generated data. Velocity in big data is the capacity to garner information or value in real-time from large volume of continuous data from different sources. This continuous high volume big data is also called *Fast Data* and is defined by Baer (2013) as a subset of big data as it encapsulates the velocity characteristics of big data. Variety denotes the dissimilar formats in which data is generated. Data could be in raw, semi-structured and structured formats (Baer, 2013; Katal et al., 2013; Kim et al., 2014; Tyagi et al., 2015; Hashem et al., 2015; Landset et al., 2015).

The classic 3V definition of big data has been extended due to explosion of the social media to include other Vs. An example is *veracity*. Veracity is concerned with uncertainty in user generated data (Ularu et al., 2012; Jin et al., 2015).

The era of big data brought about the need for drastic revision and improvement in data processing as traditional, relational data management technologies could not fulfil the performance requirements of big data (Gandomi & Haider, 2015). The limitation prompted research from academia, government and industry over the last decade. As a result, technologies are being developed to practically improve big data processing. Most of these technologies are open-sourced. An example is the *MapReduce*.

MapReduce, initiated by Google, is a programming paradigm for concurrent and distributed processing of big data across multiple servers called clusters or grid with high fault tolerance (Lee et al., 2012). MapReduce fundamentally breaks down a big task into smaller tasks and processes them in parallel. The open-source Apache Hadoop supports the MapReduce paradigm. Hadoop is a highly scalable framework used for processing big data across several machines (Apache Software Foundation, 2016). Apache Hadoop infrastructure comprises two main modules namely Hadoop Distributed File System (HDFS) and MapReduce. The purpose of HDFS is to provide fast and distributed access to data (Fan & Bifet, 2013). According to Ularu, Puican, Apostu, & Velicanu (2012), Hadoop is now the effective standard framework for processing big data.

A notable amount of data is generated as data stream and require immediate processing to extract relevant value in real-time. The importance and benefits of analysing continuous stream of data in real-time cannot be understated (Lorentz, 2013). Organizations need to gain insight from big data so that information such as opportunities, threats and performances are quickly spotted (Russom, 2013). However, the MapReduce paradigm implemented in the majority of big data technologies was designed to handle high throughput with less attention to the velocity characteristics of big data (Hashem et al., 2015). Liu, Iftikhar, & Xie (2014) also emphasized that Hadoop lacks adequate support for real-time data processing and current algorithms are ineffectual in the analysis of big data. Thus, the need for technologies to handle big data streams. This birthed Distributed Stream Processing Engines – DSPEs (Gedik et al., 2008). DSPEs process continuous volatile high-speed data as it arrives and provides approximate responses using probability. Some examples are Apache Storm and S4 (S. Chen et al., 2014).

There are other technologies that coexist with the MapReduce and streaming model to handle big data processing. It is worthwhile to mention Zookeeper (a server enabling very reliable distributed harmonization), and NoSQL databases (for handling scalability in data storage and distributed data management). These state-of-the-art technologies form the software ecosystem for big data applications and have drastically improved the capturing, storing and analysing of big data. However, these technologies exist independently to solve specific big data problems. The streaming model is unsuitable for static big data, and the MapReduce paradigm is ineffective with fast big data stream.

It is important to note that in order to provide insight and make sense out of big data, artificial intelligence and machine learning are used in the development of applications that empower computers to learn and identify complex patterns and knowledge hidden in data and automatically make intelligent predictions based on the data (Brown et al., 2011; Kraska et al., 2013).

A fundamental research area in machine learning that addresses uncertainty in data is called probabilistic reasoning. Probabilistic reasoning has proven useful in handling the veracity characteristics of big data (Dobre & Xhafa, 2014; Ghahramani, 2015). There are automated applications that use probabilistic reasoning to process data. These applications are known as Probabilistic Reasoning Systems (Zadeh, 2003).

Probabilistic reasoning systems use a probabilistic model and inference algorithm to perform computation on data. A probabilistic model is developed using Bayesian or Markov networks. This is usually a difficult task and requires extensive knowledge in these networks (Dobre & Xhafa, 2014). Again modelling real-life scenario as a probabilistic model is complex as a result of the difficulty (Sampson, 2015; Roy, 2018). This led to the concept of Probabilistic Programming.

Probabilistic programming makes it easier to develop complex probabilistic models using the powerful features of a programming language in probabilistic modelling.

1.3 Research Problem Statement

The current software ecosystem of big data technologies is appropriate for processing huge amount of both historic and real-time data independently, providing distributed processing across several servers. On one hand, the parallel and distributed batch computing is suitable for processing large volume of historic data. On the other hand, distributed stream processing frameworks tackle processing of big data streams. Again, data from diverse sources includes inconsistences and is often incomplete, thus introducing data uncertainty. Nevertheless, decision makers most often require analysis of real-time events (fast data stream) based on specific relevant history or experience (static big data) using probabilistic reasoning to enhance apt decision and meaningful action especially in times of uncertainty. Thus, a fully automated real-time probabilistic reasoning system to process both static and fast big data is imperative (Asrtikis et al., 2012; Fan & Bifet, 2013; Jagadish et al., 2014; Twardowski & Ryzko, 2014; Tyagi et al., 2015; Bhadani & Jothimani, 2016; Qiu et al., 2016).

1.4 Research Questions

Following the research problem statement, the principal research questions (RQs) are stated as follows:

RQ 1: What are the existing real-time big data solutions developed using probabilistic programming?

RQ 2: How can low latency be achieved when processing big data (both historic and real-time) using current open-source big data processing technologies and techniques in a cost-effective way?

1.5 Research Aims & Objectives

The sections on background and research problem presented above results in the primary aim of this study. The research aim is in two-fold and is summarized as follows:

Goal 1: Find existing real-time probabilistic reasoning systems implemented using probabilistic programming that process both historic and real-time big data.

Goal 2: Develop a probabilistic reasoning big data technology using probabilistic programming to process historic and real-time big data at the same time in a cost-effective and timely manner.

To achieve this aim, a list of objectives (OBJs) is outlined as follows:

OBJ 1: Perform a review of literature to identify existing real-time probabilistic big data solutions developed using probabilistic programming.

OBJ 2: Based on the result of objective 1, design a cost-effective probabilistic reasoning system that processes both historic and real-time big data using probabilistic programming.

OBJ 3: Achieve real-time response when processing both historic and real-time big data through the effective combination of current open-source big data processing tools and technologies.

It is crucial to understand each of the objectives outlined above as they contribute to the overall actualization of the research goals.

1.6 Delineation and Assumption of the Study

This research seeks to design a system for real-time big data processing and analysing using open-source big data technologies, machine learning algorithms and commodity hardware. Thus, attempting to provide decision makers with a probabilistic score or response that will assist in the process of decision-making especially in uncertain circumstances. Furthermore, due to the design characteristics of this research, this study will show a practical example on how DSR methodology is used in information technology research.

This is a technical research with very little or no fieldwork, thus there are assumptions on deployment and implementation environment. This study also assumes the availability of basic big data hardware infrastructure.

1.7 Research Methodology

One of the principal goals of this research is to develop a scalable software library that will efficiently process big data and produce a probabilistic score in real-time. In order words, this research will produce an artefact. Thus, this study will use a pragmatic approach based on design research paradigm as supported by Simon (1997).

The pragmatic approach gives researchers the flexibility to use any method or strategy corresponding to quantitative and/or qualitative research that best tackles the research problem. This provides a rich context to the study based on the complementary advantages of the pragmatic approach (Williams, 2007; Creswell, 2007; Tashakkori & Teddlie, 2010; Creswell, 2013).

Design science is considered a set of procedures used for research in technical fields such as computer science, architecture, information technology and engineering (Weber, 2010). Design science research (DSR) is technology-oriented and ventures into invention of artefacts that benefits human purposes. Hevner, March, Park, & Ram (2004) defined design as a series of actions or processes that results in a novel artefact or product. There are four potential outputs namely constructs, models, methods and instantiation, and two processes: build and evaluate in design science (March & Smith, 1995; Peffers et al., 2007).

Hevner et al. (2004) provided a framework for design science comprising environment (people, organisation and technology), knowledge base (theoretical foundations and methodology), and the specific research. These three elements are influenced by relevance and rigor. People, businesses and technologies form the problem space and present specific business needs that make the research applicable.

Furthermore, A. Hevner & Chatterjee (2010) and A. R. Hevner et al. (2004) proposed seven guidelines in design science research as follows:

1. Design as an artefact.
2. Problem relevance.
3. Design evaluation.
4. Research contribution.
5. Research rigor.
6. Design as a search process.
7. Communication of research.

These seven principles describe the process of conducting and evaluating research process in design science. In addition to the guidelines, Peffers et al. (2007) proposed a methodology applicable to design science research. This study will use the methodology described by Peffers et al.,(2007).

1.8 Organisation of the Thesis

Work done during this research is arranged and reported in seven chapters as explained below.

In the first chapter, a brief introduction expounding the context of the study is presented. The background to this study, the research problem, as well as the aims and objectives of this research are all contained in Chapter one. Furthermore, the research questions and scope of this study were also presented.

The second chapter presents background knowledge and theories associated with this research. It started with a brief introduction into the concept, history, definition and processing paradigms of big data. Chapter 2 also presents discussions around Lambda architecture, probabilistic reasoning, probabilistic reasoning systems and probabilistic programming. Chapter 2 ends with a systematic literature review highlighting the applicability of this research.

Chapter 3 is a discussion on design science research as the chosen research methodology for this study. The chapter begins with a brief explanation of research, research methodology and techniques then highlight the relevant methodology suitable for this research with backing reasons. According to the nature of this research, the design science research methodology was selected as a research methodology for this research.

In chapter 4, findings gathered from background knowledge, theories and the systematic literature review presented in chapter 2 were used to present the design concept of this research. The design concept was constructed in line with the design science research methodology discussed in chapter 3. The foremost contribution of the design concept is highlighted in chapter 4.

The fifth chapter presents a demonstration and assessment of the design concept described in chapter 4. The usefulness of the design presented in chapter 4 was demonstrated through the selection of a problem domain, then the development of a case study. The tools and technologies used in the development were described. Finally, chapter 5 presents the evaluation of the developed artefact in a simulated environment using real data.

Chapter 6 contains an evaluation of the research activities used in the advancement of the design concept presented in chapter 4. Each step of the research is measured against the documented authority to verify the validity of this research project as a design science research project.

The last chapter, chapter 7 presents a summary of each chapter, then revisits the aims and objectives of the research to present research findings and answers to the research questions listed in chapter 1. A summary of the research work is presented in chapter 7. Chapter 7 ends with the shortcomings of this study and potential subsequent research directions.

CHAPTER TWO
THEORY, BACKGROUND AND REVIEW

This chapter presents and reviews subjects that form the background theory for this thesis. This helps readers that are not familiar with the topics around this research to have an idea of what is necessary to comprehend the work presented in the later chapters of this thesis.

The first topic presented is discussions around big data in Section 2.1, and then Lambda architecture is explored in Section 2.2. Section 2.3 presents a brief description of probabilistic reasoning and systems that use the concepts as well as the emerging probabilistic programming idea. A systematic literature review focusing on probabilistic reasoning using probabilistic programming and big data is presented in Section 2.4. Finally, Section 2.5 summarizes this chapter.

2.1 Big Data

What is big data? First let us have a look at the definition of 'big'. Merriam-Webster and the Cambridge online dictionaries define the adjective 'big' as large in size, number or amount (Merriam-Webster, 2016; Cambridge Dictionary, 2016). Consequently, the initial and sometimes greater part of what many think or assert as the exact meaning of big data is towards size (Gandomi & Haider, 2015). However, if we talk about the amount, number or size of data exclusively, this gives an opinion that data has always been small until recently. This clearly is a misrepresentation. Early mass storage systems have been around to handle large data sets. An example is the IBM 3850 MSS which was used by scientists to support the 1980 United States census databases – a 'big' data at that time (Jacobs, 2009). Databases have been used to collect and store relatively large data sets for over a century. A traditional database housing data of say 500GB in size would not be considered as significantly small today. Thus, how much of data can we refer to as big? Terabytes? Petabytes? Or perhaps Exabytes? How about Yottabytes? Again, do we have to consider the volume of data exclusively to understand or properly define big data? According to Mohanty et al. (2015), this question is yet to be answered, making the expression 'big data' a misrepresentation or inappropriate label of the concept and definition of big data (Boyd & Crawford, 2011).

2.1.1 Concept of Big Data

The notion of big data seems to be ambiguous with numerous research papers and articles on big data (Li et al., 2015). It has been publicized in a variation of ways such as a contemporary variety of economic assets like gold, a marketing expression, a concise description of advancement in data technologies to better understand the world, and an insightful kit to tackle problems like crime and poverty (Lohr, 2012). These days, many people in different academic fields and professions talk about big data, even in boardrooms.

Organizations are saturated with data from burgeoning sources like clickstream, video data, sensors, etc. These data are generated by people, about people, things and how they interact and according to Davenport, Barth, & Bean (2012), organizations that utilize big data will enjoy the new capabilities and value of big data.

The epoch of big data is in progress. In sciences, big data refers to large data set that require super machines, even though there are many typical software running on desktop computers that can now process large data set. Danah Boyd & Crawford (2011) and Dannah Boyd & Crawford (2012) argued that even though the size of data generated today is quite large, volume is not the only fundamental feature of big data. As an example, data on a specific topic from social media such as Twitter is not nearly as large as prior data set not considered as big data. Again, Kitchin & Lauriault (2015) and MIKE 2.0 (2018) supported this fact, stating that not all voluminous data set are big data and some ‘small’ data set can be considered as big data.

Ward & Barker (2013) mentioned data storage and analysis as the two fundamental concepts of big data. Dannah Boyd & Crawford (2012) further asserts the idea of big data as a powerful tool to curb community ills and provide novel perception in divergent areas such as medicine, environmental sciences and terrorism.

2.1.2 Definition of Big Data

Big data is now a predominant term used in many industries and academia to illustrate a broad range of ideas in data science. As noted by De Mauro, Greco, & Grimaldi (2015), there are multiple definitions of big data because of the constant use of the term in different context and its rapid, inconsistent evolution. These definitions of big data are in many instances, divergent thus creating ambiguity in discourse relating to big data – some define big data on what it is, others define big data based on what it does (Gandomi & Haider, 2015).

One of the most common definitions of big data can be linked to a Meta (now Gartner) report by Laney (2001). Laney expressed an increase in the *volume* of data, *variety* of data and *velocity* at which data is acquired or generated. This postulation, popularly nicknamed *3Vs* is associated with the concept of big data. Although Laney made no direct reference to big data in the report, his assertion has been used as a key composition to the definition of big data (Ward & Barker, 2013; Gandomi & Haider, 2015). However, this model had been extended. An example is the addition of *value* as seen in the definition of big data by IDC in a paper titled *Extracting Value from Chaos:* “Big data technologies describes a <u>new</u> generation of technologies and architectures, designed to economically extract <u>value</u> from very large <u>volume</u> of a wide <u>variety</u> of data, by enabling high-<u>velocity</u> capture, discovery, and/or analysis.” (Gantz & Reinsel, 2011). More authors have added additional Vs such as *veracity* (Ularu et al., 2012; Demchenko et al., 2013; Assunção et al., 2015; Jin et al., 2015; Miloslavskaya & Tolstoy, 2016), *variability* (Fan & Bifet, 2013; Katal et al., 2013; Philip Chen & Zhang, 2014; Tyagi et al., 2015; Miloslavskaya & Tolstoy, 2016), and *visibility* (Miloslavskaya & Tolstoy, 2016). These definitions of big data outline the characteristics of big data.

Khan et al (2014) explained big data as numerous data generated, captured and processed rapidly, and difficult to classify in the traditional relational databases. In the opinion of Snijder, Matzat, & Reips (2012, p. 1), big data defines huge and complex data set that “become awkward to work with using standard statistical software.”. Similarly, Philip Chen & Zhang (2014) defines big data as an accumulation of enormous and heterogeneous data set thus making it an arduous task for traditional databases to process. These definitions of big data fall under the category of definitions that compare software tools for big data.

Data analytics is also relevant when defining big data. Dannah Boyd & Crawford (2012, p. 665) outlined “computational turn in thoughts and research” as well as tools and procedures used in the processing of big data. This class of definition highlights the influence of big data on society. The National Institute of Standards and Technology (NIST) defined big data, pointing out the architectural aspect of big data (NIST, 2018). Similarly, Oracle (Dijcks, 2012) defined big data with emphasis on infrastructure, thus presenting solutions to big data (Ward & Barker, 2013).

Another category of big data definition highlights the importance of computing power (Microsoft, 2013). This definition also introduced the concept of machine learning and artificial intelligence as related set of technologies that form a significant part of big data.

These existing definitions of big data show that the focus of big data is mainly on its characteristics (the Vs), specialised technologies and analytical methods used to process big data. Thus, De Mauro et al. (2015, p. 103) proposed a consensual definition of big data: “Big Data represents the Information assets characterised by such High Volume, Velocity, and Variety to require specific Technology and Analytical Methods for its transformation into Value.”

2.1.3 Brief History of Big Data

Some believe big data is new and different from what has been. However, Barnes (2013) stated that big data did not start with Google or Apple but has been around. Big data is a combination of different elements, each with its own history, merging at our current moment. As mentioned in Section 2.2.1, data storage and analysis are two primary concepts linked with big data. These ideas are not new and predates the present trend. Investigations on the advancement of big data indicate that research into big data started in the 1970s (Ularu et al., 2012). However, Tyagi et al. (2015) claims that big data emerged for the first time in 1998 in a book titled *Big Data and the NextWave of InfraStress* by John Mashey and subsequently, the first academic paper on big data was in 2000 by Diebold. In 2012, the government of the United States of America publicized a national policy titled *Big Data Research and Development Initiative* to support education, collaboration and research into big data.

The history of big data is generally connected with the evolution of efficient storage and data management systems with respect to data size. Han Hu, Yonggang Wen, Tat-Seng Chua, & Xuelong Li (2014) divided the history of big data into four stages with respect to data volume. The first stage is the *Megabyte to Gigabyte* stage.

Megabyte to Gigabyte stage occurred between the late 1970s and early 1980s where the need to store data and perform analysis and reporting became apparent. This resulted in the database machine concept which involves specialized hardware and software integration to accumulate and examine data. Digital technology became more publicized in the late 1980s causing the volume of data to increase to terabytes. Database machines became insufficient to effectively store and manage data. This led to the *Gigabyte to Terabyte* stage where the “share-nothing” idea was suggested.

The share-nothing framework is made up of a group of database systems running on a networked cluster (DeWitt & Gray, 1992), each with its separate memory, processor and disk. According to Borkar et al. (2012) and M. Chen et al. (2014), Teradata Corporation developed the first commercialised database system based on the share-nothing architecture. Parallel database systems improved data storage and processing performance and thus the idea was welcomed. However, during the late

1990s, a boost in the use of Internet increased the size of data to petabytes and introduced unstructured or semi-structured data. Parallel databases could not effectively handle unstructured data even though they were suitable for structured data. Thus, internet companies like Yahoo and Google were faced with the challenge of indexing and querying the rapidly growing content of the web created by users. This was the *Terabyte to Petabyte* stage.

To tackle this challenge, Google developed the Google File System (GFS) to collect large data set, and a programming model called MapReduce that handles the processing of large data set (Dean & Ghemawat, 2008). GFS is a reliable, fault-tolerant and scalable distributed file system that runs on thousands of commodity hardware (Ghemawat et al., 2003). Yahoo and Facebook created the open-source version of MapReduce and GFS called Hadoop and Hadoop Distributed File System (HDFS) (Borkar et al., 2012). The multiplication of data sources such as sensors and mobile devices in the mid-2000s resulted in a deluge of data in different formats such as audio, video, files and images, mostly referred to as semi-structured and unstructured data. This required a new paradigm to effectively manage and process large-scale semi-structured and unstructured data. Thus, the NoSQL databases were revealed. Again, giant technology companies like Amazon and Google implemented their versions of NoSQL called Dynamo and Big Table respectively (Borkar et al., 2012).

The fourth stage in the history of big data is called the *Petabyte to Exabyte* stage. Data sources have continued to increase since year 2000. Han Hu et al. (2014) predicted that the volume of generated data will continue to multiply, stating that no technology has been developed to handle larger data set. However, current technologies can handle terabyte to petabyte of data.

2.1.4 Big Data Processing

Big data require techniques and tools to capture, organize and analyse it to derive meaningful value. According to Philip Chen & Zhang (2014), these tools and techniques are developed using a combination of knowledge from different specialties like Mathematics, Computer Science and Statistics. Labrinidis & Jagadish (2012) explained five steps involved in the process of mining value from big data. These steps are further categorized under two main processes – *data management* and analytics, as shown in Figure 2.1.

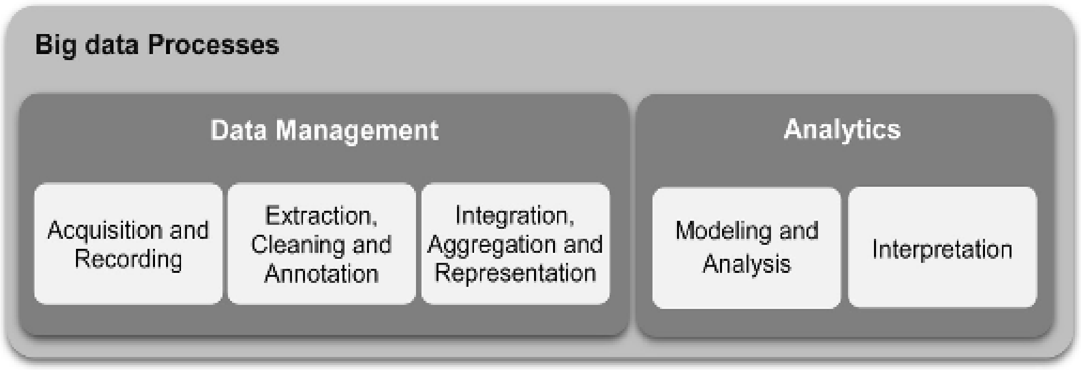

Big data Processes
Data Management
Acquisition and Recording
Extraction, Cleaning and Annotation
Integration, Aggregation and Representation
Analytics
Modeling and Analysis
Interpretation

Figure 2.1: Big data processes

(Gandomi & Haider, 2015: 141)

Data management is concerned with the tools, processes and technology used to gather, store and prepare data for the analysis phase (Chen et al., 2013). In the analysis stage, analysis algorithms are used to examine the data. This process is referred to as *big data analytics*. According to Assunção et al. (2015) and Delen & Demirkan (2013), big data analytics tools are grouped as *descriptive, predictive* and prescriptive (see Figure 2.2). Descriptive analytical tools discover patterns from historical data by modelling past actions (Fitz-enz, 2009). Predictive analytics uses statistical models and machine learning algorithms on both past (historic) and current data to attempt to forecast future trends (Shmueli & Koppius, 2010; Siegel, 2013; Zakir et al., 2015). Prescriptive analysis calculates actions and their corresponding influence on business activities using optimization (Evans & Lindner, 2012; Song et al., 2013).

Contemporary big data analytical tools mainly process big data in batch or stream (Barlow, 2013; Huang & Liu, 2014; Jambi & Anderson, 2017). Early solutions developed to process big data were based on batch processing. Batch processing is used to process very large volume of historic or static data, that is, data that has been collected and stored over time (Adhianto et al., 2010). Most batch processing tools were implemented using the MapReduce framework (Shahrivari, 2014).

The MapReduce framework comprises three components namely a distributed file system, a distributed NoSQL database and a MapReduce engine. The MapReduce engine provides a simple and effective programming model that offers parallel and distributed processing of large data set on clusters of commodity hardware. This programming model is highly scalable and fault-tolerant. MapReduce provides two elementary functions – *map* and *reduce*, that allows users to implement computation on big data. The input to the MapReduce engine is a list of key-value pairs. The map function performs computation on the input to produce a set of zero or more intermediate key-value pairs. Subsequently, all intermediate values corresponding to an intermediate key are grouped and passed to the reduce function. The reduce function iterates through each intermediate key and its associated list of values and performs computation on each to produce results. Each reduce iteration usually produces zero or one output (Dean & Ghemawat, 2008; Dean & Ghemawat, 2010; Lee et al., 2012; Shahrivari, 2014; Bhadani & Jothimani, 2016). Lämmel (2008, p. 1) summarized the map-reduce process in five basic notions as follows: "*(i) iteration over the input; (ii) computation of key/value pairs from each piece of input; (iii) grouping of all intermediate values by key; (iv) iteration over the resulting groups; (v) reduction of each group*".

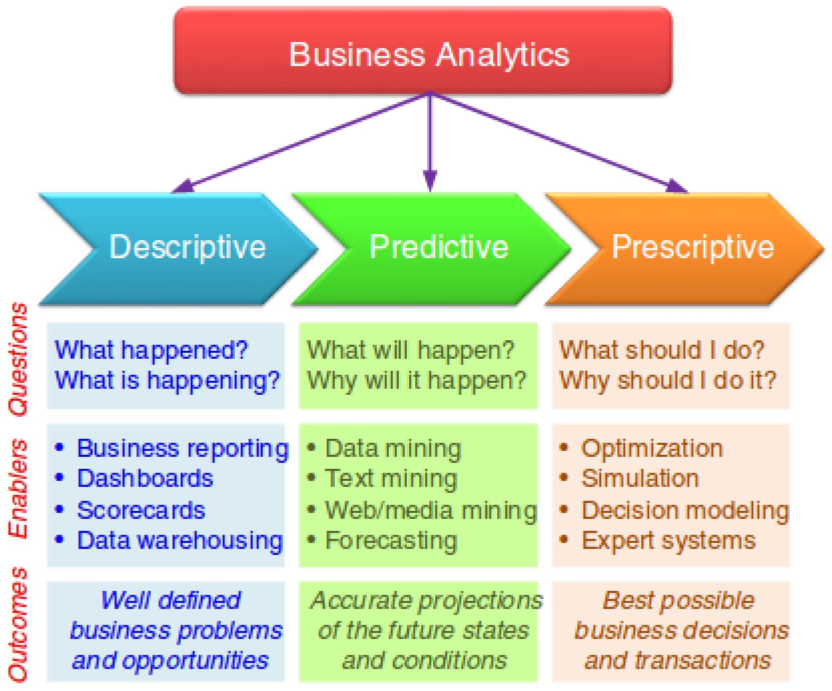

Business Analytics
Descriptive
Predictive
Prescriptive
Questions
Enablers
Outcomes
What happened? What is happening?
What will happen? Why will it happen?
What should I do? Why should I do it?
• Business reporting • Dashboards • Scorecards • Data warehousing
• Data mining • Text mining • Web/media mining • Forecasting
• Optimization • Simulation • Decision modeling • Expert systems
Well defined business problems and opportunities
Accurate projections of the future states and conditions
Best possible business decisions and transactions

Figure 2.2: Categories of analytics

(Delen & Demirkan, 2013: 361)

Apache foundation developed an open-source implementation of MapReduce framework, called Apache Hadoop, although Hadoop Started in Yahoo! (Bifet, 2013). Hadoop has two core components – HDFS and HBase (Fan et al., 2014). HDFS is a replacement of Google's GFS while HBase is used instead of BigTable data store. There are other MapReduce implementation such as DISCO, however, according to Lee et al. (2012), Lin, Leu, & Chen (2015) and Xindong Wu et al. (2014) Hadoop is more popular. Hadoop has been extensively used and widely accepted as the standard for big data processing in academia and industry (S. Chen et al., 2014; Liu et al., 2014; Raghupathi & Raghupathi, 2014; Lin et al., 2017; Ramírez-Gallego et al., 2018). The MapReduce paradigm is considered the first generation big data processing framework (Adhianto et al., 2010; Nair et al., 2017), and according to Adhianto et al. (2010), Hadoop marked the end of the first generation big data processing framework as shown in Figure 2.3.

Hadoop and MapReduce framework presents advantages such as scalability, ease of use, flexibility and fault-tolerance. Despite these advantages, MapReduce and Hadoop have some limitations (Khan et al., 2014; Bhadani & Jothimani, 2016). One of such is the high latency when processing data due to its batch processing nature (Taxidou & Fischer, 2013; Sagiroglu & Sinanc, 2013; Wu et al., 2015; Vakali et al., 2016; Mohapatra et al., 2016; Yang et al., 2017).

The long processing time associated with batch processing became unbearable to end-users. Users usually require response to queries to be in near real-time or real-time. Few examples can be seen in the case of crisis management, surveillance and the stock market where decisions need to be taken as quickly as possible based on results from processing events (Perera & Suhothayan, 2015). Thus, stream processing on big data became inevitable and birthed the second generation of big data processing frameworks (Adhianto et al., 2010; Gebara et al., 2015; Bajaber et al., 2016; Nair et al., 2017).

Stream (real-time) processing enables scalable computation on big data stream (Hirzel et al., 2017). This handles the velocity characteristics of big data and ensures low latency by processing small chunks of data (Adhianto et al., 2010; Perera & Suhothayan, 2015; Wang et al., 2016). According to Strohbach, Ziekow, Gazis, & Akiva (2015), data velocity means fast flowing data that must be processed in a negligible amount of time. Adhianto et al. (2010) explained that the concept of stream processing is closely related to that of batch processing. In stream processing, batch processing is done on small chunks of data stored in memory instead of a secondary data store. However, instead of one-time queries to stored data as in the case of batch processing, stream processing enables continuous evaluation of queries on new data to produce new responses (Margara et al., 2014).

To tackle the high latency in batch processing of big data, many open-source stream processing platforms were developed. Some examples are Apache Storm (Apache Software Foundation, 2015), Apache Samza (ApacheSamza, 2016), SQLstream (SQLstream, 2017), Apache Spark (Zaharia et al., 2010; Zaharia et al., 2016), Apache S4 (Apache Software Foundation, 2010), Apache Flume (ApacheFlume,

2016) and Apache Kafka (ApacheKafka, 2017). These processing frameworks enable real-time analysis and fast response to facilitate real-time decision making (Xindong Wu et al., 2014).

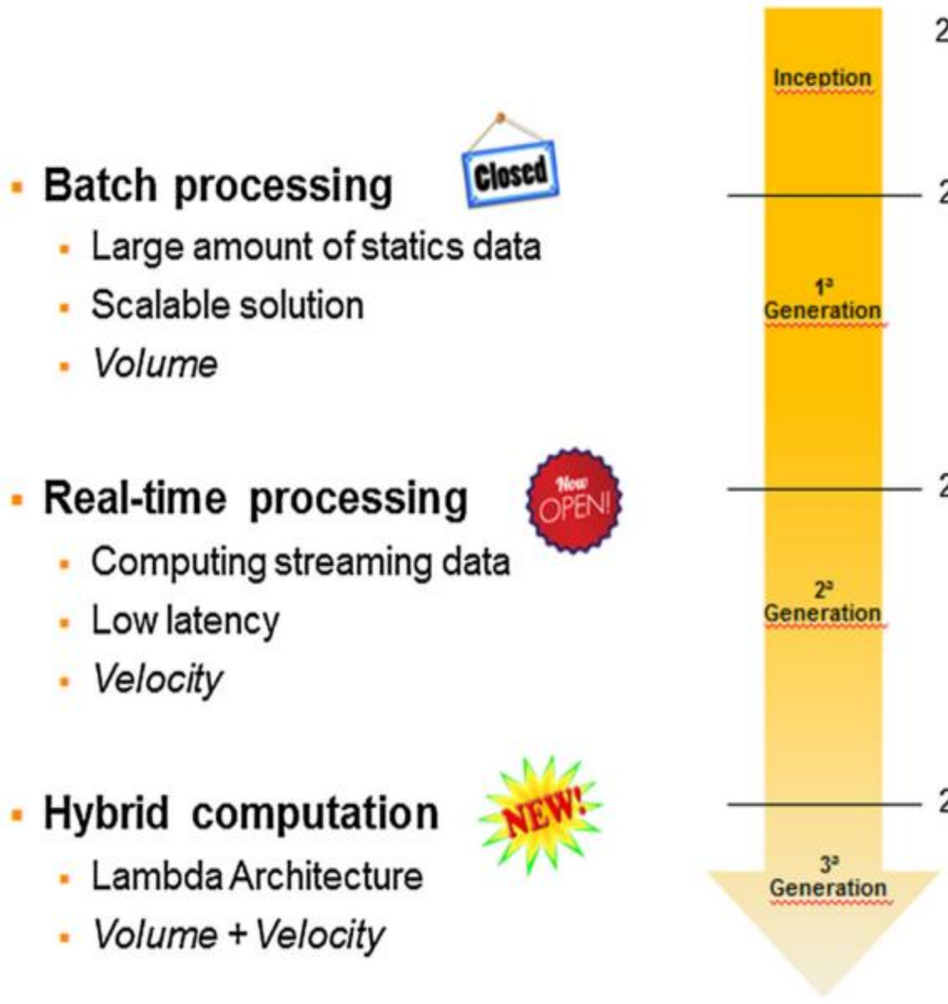


Figure 2.3: Data processing paradigm

(Adhianto et al., 2010: 2081)

Although stream processing provides low latency, a major pitfall or stream processing framework is that they do not output accurate responses as compared to batch processing platforms (Yang et al., 2017). Thus, using stream processing framework to replace batch processing would be insufficient in handling the problems of big data.

A single tool or technique may not serve as a panacea for all big data problems. The combination of batch and stream processing in one big data platform could suffice. Again, decision makers often need to make decisions based on historic (batch) data and real-time data (Twardowski & Ryzko, 2014; Kiran et al., 2015). Some researchers such as Adhianto et al. (2010), Jambi & Anderson (2017) and Zhou, Simmhan, & Prasanna (2013) endorsed the need to support batch and stream processing in a big data platform. This structural combination is regarded as the hybrid computation (Miloslavskaya & Tolstoy, 2016) or third generation of big data processing framework (Adhianto et al., 2010). See Figure 2.3.

The need for a hybrid big data platform sparked research and resulted in models such as the Kappa architecture designed by (Kreps, 2014), the Liquid architecture developed by (Fernandez et al., 2015), and Lambda architecture introduced by (Marz & Warren, 2015).

2.2 Lambda Architecture

Marz & Warren (2015) proposed a new model called *Lambda Architecture* to handle large data set in real-time. This model supports the idea that neither batch nor stream processing alone could handle all big data problems. Thus, instead of a single technology, Lambda architecture explains the combination of different big data tools/technologies to provide a generic solution. According to Marz & Warren (2015), the architecture presents a common model to implement computation on arbitrary data set using arbitrary functions in real-time. Lambda architecture also describes a guideline that assists developers/designers in choosing the right technology to combine, and how to combine them (Twardowski & Ryzko, 2014).

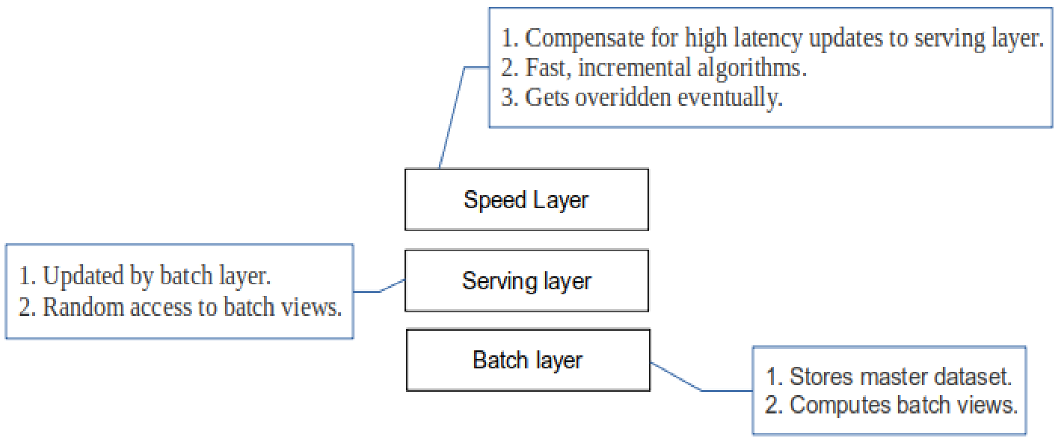

1. Compensate for high latency updates to serving layer.
2. Fast, incremental algorithms.
3. Gets overidden eventually.
Speed Layer
1. Updated by batch layer.
2. Random access to batch views.
Serving layer
Batch layer
1. Stores master dataset.
2. Computes batch views.

Figure 2.4: Lambda architecture layers

(Adopted from Marz & Warren, 2015)

Astakhov & Chayel (2015, p. 4) defined Lambda architecture as “a data-processing design pattern to handle massive quantities of data and integrate batch and real-time processing within a single framework”. Köhler, Kaniovskyi, & Benkner (2015) also defined Lambda architecture as a general-purpose blueprint to develop scalable and fault-tolerant big data systems. In the words of Vögler, Schleicher, Inzinger, & Dustdar, (2017, p. 5), Lambda architecture is “a generic, scalable, and robust data processing system, specifically designed to serve massive workloads and a wide range of use cases”. Thus the architecture promises scalability, extensibility, generalization and fault-tolerance (Zheng et al., 2017). Ganchev, Ji, & O’Droma (2016), Jambi & Anderson (2017) and Twardowski & Ryzko (2014) claims that Lambda architecture is widely used in academia and industry.

Lambda architecture proposes a three-layered big data system. As stated by Marz & Warren (2015), each layer handles a specific big data problem and is built on top of the functionality of the layer beneath it as shown in Figure 2.4. Data flows into the architecture and is concurrently added to and processed by the batch and speed layers.

The batch layer represents the core of the Lambda architecture. This layer acts as an immutable append-only data repository of unprocessed data. Here, periodical computational jobs usually implemented using a batch processing framework, processes the raw data (also known as the master data set) and produces batch views which are sent to the serving layer for queries. New data are stored in the batch layer and included in the next batch computation cycle. Results from batch computations are comprehensive and more accurate than those from the speed layer (Harrison, 2014; Hasani et al., 2014; Villari et al., 2014; Astakhov & Chayel, 2015; Tseng et al., 2016; Yang et al., 2017).

Real-time processing happens in the speed layer to balance the long-running batch job in the batch layer. Incoming data into the system is processed immediately in the speed layer. This enables real-time analysis using incremental model to produce up-to-date real-time views. Results from the real-time processing are usually approximations. Thus the real-time views are repeatedly discarded as soon as computation is done on the same data in the batch layer (Hasani et al., 2014; Villari et al., 2014; Kiran et al., 2015; Astakhov & Chayel, 2015; Yang et al., 2017).

The serving layer presents an integration of batch views and real-time views for queries (Yang et al., 2017; Jambi & Anderson, 2017). According to Astakhov & Chayel (2015), Hasani et al. (2014) and G. Liu, Zhu, Saunders, Gao, & Yu (2015), this layer is updated with batch views from the batch layer and enables fast and random access to the batch views when needed.

The combination of the speed and serving layers ensures low latency results that include computations on both batch data and real-time data (Huang & Liu, 2014; Liu et al., 2014; VANHOVE et al., 2016).

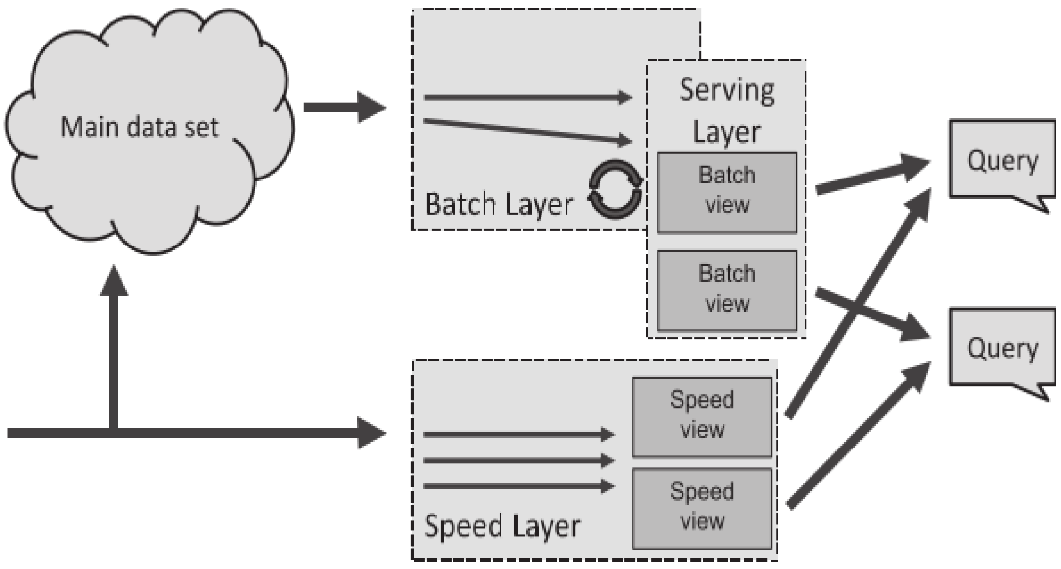


Figure 2.5: Lambda architecture

(VANHOVE et al., 2016: 298)

In summary, Lambda architecture can be mathematically expressed as follows:

```
batch view = function(all data)
real-time view = function(real-time view, new data)
query = function(batch view. real-time view)
```

Figure 2.5 shows a conceptual overview of all the layers of Lambda architecture.

2.3 Probabilistic Reasoning

As a matter of fact, all humans are decision makers. According to Kutty, Kumar Shee, & Pathak (2007), decision-making is the process of identifying and selecting the best option from a range of alternatives. All our actions are because of some decision. Decision-making is usually based on gathered information, values, beliefs and preferences.

In most cases, making decisions are usually straightforward. Some examples are deciding on a time to rest, or a specific food to eat or stopping your vehicle at a stop sign. On the other hand, some decision-making involves complex processes usually based on many sources of evidence (Yang & Shadlen, 2007). This is seen in the case of uncertainty where decisions are made based on the likelihood of unknown or pending events such as the outcome of a medical diagnosis. According to (Tversky & Kahneman, 1975), answers with respect to the likelihood of events are conveyed in statements that typically starts with phrases such as “Suppose that…”, “In anticipation…”, “It is likely…”, “Usually…”, etc. In other words, these responses are in the form of probabilities (Zadeh, 2003).

The activity associated with decision-making that involves thinking and logical argument is known as reasoning (Merriam-Webster, 2018; Collins Dictionary, 2018). In certain situations, probability is used to express the degree to which an event is possible. Thus, probabilistic reasoning uses logic and probability to make decision-making easier in times of uncertainty.

Probabilistic reasoning also known as uncertain reasoning or probabilistic logic is the combination of probability theory and deductive logic to benefit from formal argument in uncertain situations (Szolovits & Pauker, 1978; Haenni, 2005; Luger & Chakrabarti, 2008; Alon, 2013). In the same manner, Pfeffer, (2016) defined probabilistic reasoning as the union of information/understanding of a particular situation with the laws of probability to discover hidden details that could be important in decision-making.

According to Gonzalez (2012), probabilistic reasoning defines the background of modern statistics and machine learning and is used to model noisy data, interpret complicated situations and express uncertainty.

Today, big data is gathered from different sources with disparate degree of consistency. This introduces errors and incomplete data that must be handled (Jagadish et al., 2014). Probabilistic reasoning or modelling is usually suitable for managing data uncertainty (Adar & Re, 2007; Chen et al., 2013; Wampler, 2013; Bendler et al., 2014; Ghahramani, 2015).

2.3.1 Probabilistic Reasoning System

Decision-making has become a mathematical science (Figueira et al., 2005). This has provided a formal thinking process expressed in mathematical terms to enable transparent and better decision-making (Saaty, 2008). This is also known as *Probability Theory* (Ghahramani, 2015).

Decision-making processes in many fields such as fraud detection, computer vision, data mining and weather forecasting have been automated. According to Zadeh

(2003), these automated applications use probabilistic reasoning and are referred to as *probabilistic reasoning systems*.

A probabilistic reasoning system comprises two main modules namely *Probabilistic model* and *Inference algorithm* (Pfeffer, 2016). See Figure 2.6. The probabilistic model is an encoding of comprehensive knowledge and relevant factors about a specific field in quantitative, probability theories such as Bayesian networks also called belief networks (Liu et al., 2010), hidden Markov models (Rabiner, 1989; Durbin et al., 1998) and stochastic grammar (Manning & Raghavan, 2009; De Raedt & Kersting, 2003; Ábrahám & Havelund, 2016; Williams, 2018).

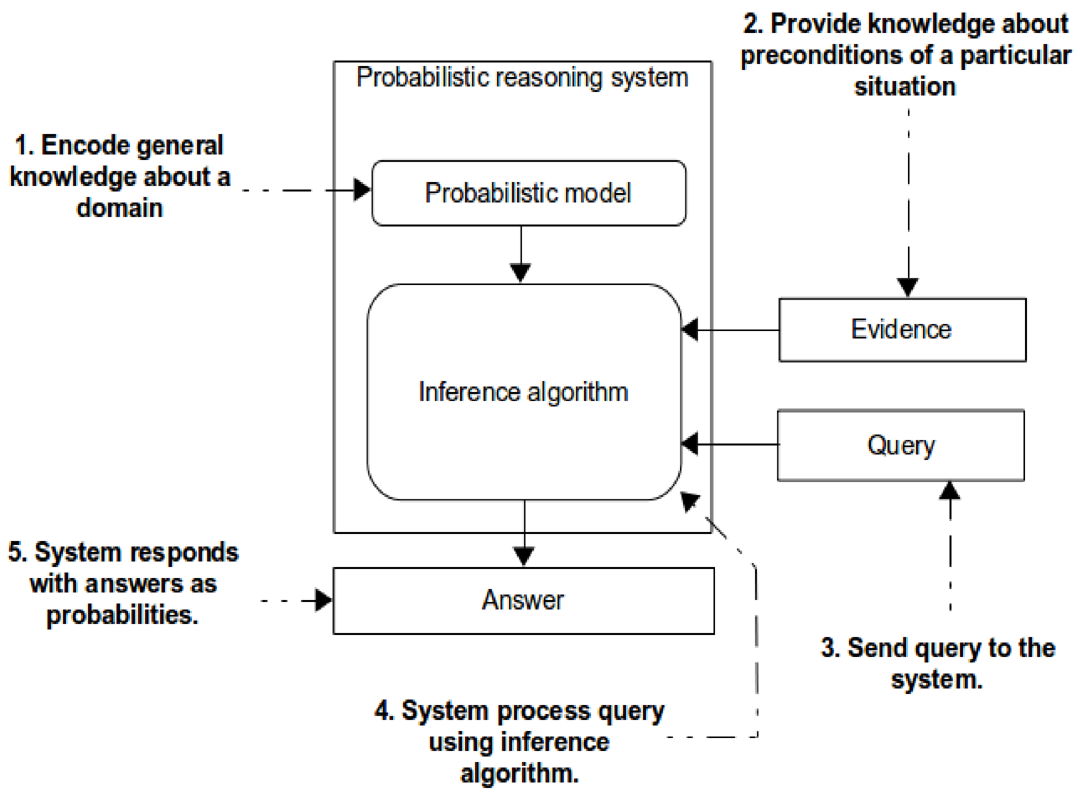

2. Provide knowledge about preconditions of a particular situation
Probabilistic reasoning system
1. Encode general knowledge about a domain
Probabilistic model
Inference algorithm
Evidence
Query
5. System responds with answers as probabilities.
Answer
3. Send query to the system.
4. System process query using inference algorithm.

Figure 2.6: Mechanism of a probabilistic reasoning system

(Adopted from Pfeffer, 2016: 6)

A specific information or fact about a situation in the domain is presented to the probabilistic reasoning system along with a property of the situation that needs to be determined. This fact is known as evidence. The inference algorithm uses the probabilistic model and the given evidence to provide response to queries as probabilistic score. This process is called *probabilistic inference*. The probabilistic model, evidence and responses to queries are all connected mathematically by the laws of probability (Pfeffer, 2016).

A probabilistic reasoning system can be used to predict future happenings, understand or deduce the cause of an event, learn from previous events to improve prediction and general knowledge of a domain. Just like any other machine learning system, a probabilistic reasoning system will produce accurate prediction based on the size of data (Ghahramani, 2015). Thus, prediction quality of a probabilistic reasoning system depends on how close the probabilistic model represents the real-world situations and the amount of data provided. Some examples of a probabilistic reasoning system are BayesiaLab (Conrady & Jouffe, 2013; Bayesia, 2018) and Netica (Norsys, 2013).

2.3.2 Probabilistic Programming

Probabilistic models in all probabilistic reasoning systems are expressed using *representation language*. A representation language is used to encode general knowledge of a domain in a probabilistic model. Bayesian networks and hidden Markov models are some examples of representation languages. These representation languages determine or influence the type of probabilistic model a probabilistic reasoning system can manage. The capability of a representation language to encode diverse knowledge in its models is known as the *expressive power* of the representation language (Pfeffer, 2016).

Designing a probabilistic model involves a combination of mathematical constructs, pseudo codes and natural language. This is usually an arduous task that requires extreme technical expertise (Luger & Chakrabarti, 2008; Dobre & Xhafa, 2014; Zhang et al., 2014). As a result, it is difficult to model many real-life circumstances, and thus expensive (Pfeffer, 2009; Sampson, 2015; Pfeffer, 2016; Roy, 2018). Furthermore, according to Goodman & Stuhlmüller (2014) probabilistic models are now more complicated thus they require new tools to develop and represent them. To fill this representational gap and develop new model representations, the machine learning and programming language communities started work on a research area called *probabilistic programming* (Hicks, 2014; Sampson, 2015).

The underlying concept of probabilistic programming is centred around the adoption of powerful features of programming languages in probabilistic modelling and inference (Dries et al., 2015). According to Pfeffer (2009, 2016), probabilistic programming provides a much easier procedure to express complex probabilistic models using a programming language. Thus, instead of expressing models in declarative mathematical notations such as Bayesian networks, models are represented using executable functions or procedures (Ghahramani, 2015). Probabilistic programming enables easier composition of probabilistic models and automatic inference computation on the models to handle uncertainty (Prékopa, 2003; Acharya & Biswal, 2011; Goodman & Stuhlmüller, 2014; Hicks, 2014; Wood et al., 2014; Andrew D. Gordon et al., 2014; Dries et al., 2015; Narayanan et al., 2016; Gehr et al., 2016).

Several probabilistic programming systems or languages has been developed since the conception of research in this area. Some implementations are based on functional programming languages such as Church (Goodman et al., 2008), Anglican (Wood et al., 2014; Tolpin et al., 2016), IBAL (Pfeffer, 2007), and Venture (Mansinghka et al., 2014). BLOG (Milch et al., 2007), PRISM (Sato, 2008), and Markov Logic (Domingos & Richardson, 2007) are examples of probabilistic programming systems based on logic programming. Figaro (Pfeffer, 2009; Pfeffer, 2016) is based on object-oriented and functional programming. Other examples of probabilistic programming systems are BUGS (Lunn et al., 2009), Tabular (Andrew D Gordon et al., 2014), and Stan (Carpenter et al., 2016). According to Pfeffer (2016), these probabilistic programming systems are probabilistic reasoning systems that use programming languages as their representation language.

2.4 Probabilistic Programming System and Big Data: A Systematic Review

Probabilistic reasoning is very useful in uncertainty and represents an underlying principle of machine learning. Organizations such as Amazon, Google and Microsoft use probabilistic reasoning to make sense of data resulting in various applications used for predictions, detection, diagnosis and recommendation (Pfeffer, 2016). However, as mentioned in Section 2.3.1, most of the available probabilistic reasoning systems are limited in the set of knowledge they can express in their models. This motivated research into probabilistic programming systems that combines two powerful concepts (probabilistic modelling/reasoning and programming language) to achieve easier representation of complex probabilistic and real-life situations into models.

The concept of probabilistic programming systems is quite new (Goodman & Stuhlmüller, 2014; Sampson, 2015; Pfeffer, 2016). However, researchers such as Lake, Ullman, Tenenbaum, & Gershman (2017), suggests that its potential in artificial intelligence systems is crucial. It is therefore necessary to find out if the concept of probabilistic programming has been used in big data processing. Thus, this section presents a systematic review of big data applications implemented using probabilistic programming systems. The purpose of this review is to identify available big data solutions that used the concept of probabilistic programming systems in the big data space to handle big data problems. This review follows the guideline as proposed by (Kitchenham & Charters, 2007).

2.4.1 Systematic Literature Review (SLR) Questions

In systematic literature reviews, it is important to clearly specify review question(s) in order to provide the review scope. Kitchenham & Charters (2007) proposed the adoption of the *Population, Intervention, Comparison, Outcome, Context* (PICOC) criteria used in systematic literature reviews in the field of medicine to help in the formulation of review questions.

Population is concerned with the specific group affected by the research. Kitchenham & Charters (2007) explained that in software engineering, the population could be either a distinct software engineering role, a class of software engineer, an application area or a category of industry. In this review, the population falls under an application area which is big data processing technologies published in literature.

Intervention refers to the software tools that handles the issues in the population. Probabilistic programming system for processing big data is the intervention focus of this study.

Comparison addresses the software procedure used to compare the intervention. Comparison is achieved in this study by comparing the various probabilistic programming systems in the intervention. In this study, the outcome is to discover big data processing platforms that use the advantages of probabilistic programming systems.

Finally, the context describes the circumstances or conditions of the comparison. The context of this review is academia.

The goal of this systematic review is to collect and investigate all possible and effective big data solutions that used probabilistic programming as a reasoning framework to process big data. Accordingly, with reference to the goal and PICOC criteria, the following review questions (SLRQs) were formulated:

SLRQ1	What are the existing big data applications built on probabilistic programming systems to handle big data problems?
SLRQ2	How do these solutions compare to one another?

SLRQ3	What is the strength of the evidence in support of the different solutions found in RQ1?
SLRQ4	What implication(s) will findings from RQ1 to RQ3 have on this research?

2.4.2 Review Protocol

The purpose of the review protocol is to avoid or prevent a research's bias that could negatively influence the goals and objectives of the systematic review process. Defining the review protocol is a significant step in the process of a systematic literature review (Kitchenham & Charters, 2007; Okoli & Schabram, 2010). This section specifies the methods used in this systematic review. The review protocol involves describing the search strategy, study selection criteria, quality assessment criteria, data collection, and data synthesis.

2.4.2.1 Search Strategy

The search strategy was formed by first listing all probable sources that may provide relevant literature to the systematic review. Table 2.1 shows the list of selected digital libraries used. After the selection of digital libraries, a decision on how best to search for relevant studies on the selected digital libraries was specified by identifying search terms or keywords.

Table 2.1: Sources used in search strategy

Source	URL	Researcher
ACM Digital Library	http://dl.acm.org	Arinze
IEEE Explore Digital Library	https://ieeexplore.ieee.org/Xplore/home.jsp	Kabaso
ScienceDirect	http://www.sciencedirect.com	Arinze & Kabaso
SpringerLink	http://link.springer.com/	Arinze

The second step of defining search string(s) was carried out by selecting the most relevant keywords based on the research questions and the research topic. Synonyms and different spellings of keywords were also used to formulate the search string. The selected keywords are as shown in Table 2.2. The search was performed on the digital libraries dating from January 2008 to April 2018. The digital libraries provide search functionalities where keywords or search strings can be entered. They also have the *Advanced Search* option that allows users to form search strings with conjunctions like *AND* and/or *OR*. This search took place between March and April 2018 using the final search string:

Final Search String (FSS): *(Big data processing) AND (Application OR Framework OR Software OR Infrastructure OR Platform OR Solution) AND (Probabilistic* *programming OR Probabilistic programming system OR Probabilistic programming language)*.

Table 2.2: Search terms

	Category 1	Category 2	Category 3
Phrase 1	Big data processing	Application	Probabilistic programming
Phrase 2		Framework	Probabilistic programming system
Phrase 3		Software	Probabilistic programming language
Phrase 4		Infrastructure	
Phrase 5		Platform	
Phrase 6		Solution	

Paper titles, keywords and abstract formed the basis of this search. To arrive at the search string, three categories of search terms were used (see Table 2.2). Category 1 contains synonyms of the same word with similar meaning within big data literature. Category 1 finds all literature on big data processing. Category 2 retrieves all studies on software systems while category 3 consist of synonyms of the same word with similar meaning within probabilistic programming literature and retrieves all research related to probabilistic programming. An intersection of the three categories yields the search string that helped to find relevant studies needed in this systematic review.

Table 2.3: Modified search string according to specific library requirement

Digital Library	Modified Search String
ACM Digital Library	(+Big +data +processing application framework software infrastructure platform solution +probabilistic +programming system language)
SpringerLink	big AND data AND processing AND (Application OR Framework OR Software OR Infrastructure OR Platform OR Solution) AND “Probabilistic programming” OR “Probabilistic programming system” OR “Probabilistic programming language”

However, each digital library implemented the *Advance Search* functionality differently, thus to retrieve all the relevant literature from each library, the final search string was adjusted for ACM Digital and SpringerLink as shown in Table 2.3 to allow for their specific requirement. The other 3 libraries used the FSS as is. A total of 293 results were found. See Table 2.4.

Table 2.4: Search string execution result

Source	Number of Studies
ACM Digital Library	3
IEEE Xplore Digital Library	4
ScienceDirect	0
SpringerLink	286
Total	**293**

2.4.2.2 Study Selection

The selection of relevant or primary studies is needed to filter out studies that are not related to the present research. First, studies not in the computer science and information technology domain were ignored, then duplicated studies and studies with titles that are clearly not relevant to the review were eliminated. Papers in any other language other than English were removed. Again, studies published before January 01, 2008 and papers that are not conference proceedings and journal articles were disregarded. After this initial filtering, 3 studies were left. To further filter out irrelevant studies, a set of inclusion/exclusion and quality screening criteria were developed. The criteria were carried out in two phases:

1. Abstract (and keyword) inclusion/exclusion evaluation.
2. Full text inclusion/exclusion evaluation.

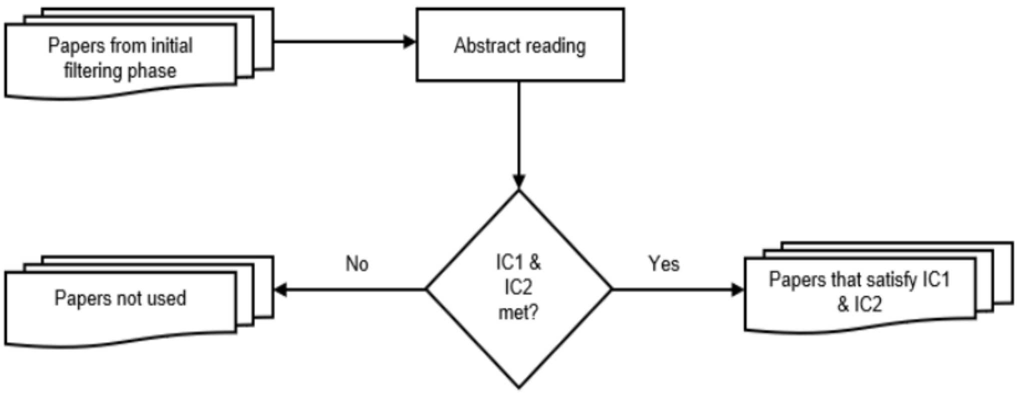

Papers from initial filtering phase
Abstract reading
No
IC1 & IC2 met?
Yes
Papers not used
Papers that satisfy IC1 & IC2

Figure 2.7: The abstract inclusion/exclusion evaluation process

In the abstract evaluation phase illustrated in Figure 2.7, the abstracts (and keywords) of the 3 papers from the search stage were read. The following inclusion/exclusion (IC) criteria were used to select primary studies:

IC1 The main concern of the study is on big data processing using the probabilistic programming concept. Therefore, the terms “Big data” or “Big data processing” and “Probabilistic programming” or “Probabilistic programming system” or “Probabilistic programming language” must be mentioned in the title, keywords or abstract of the paper.

IC2 The paper is a primary study and represents empirical results.

The two researchers examined all 3 studies individually and carefully screened them based on IC1 and IC2 to make sure that possible relevant papers were not rejected.

All disagreements were discussed and reconciled. After the abstract filtering, all 3 studies were rejected.

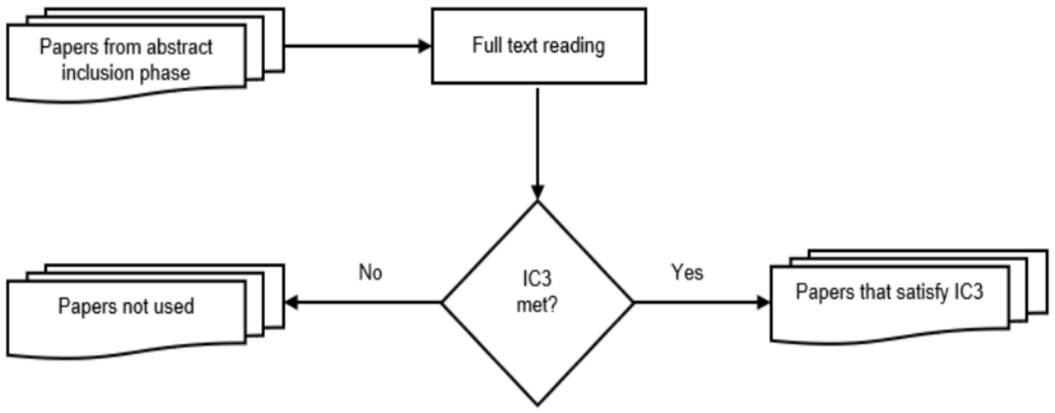


Figure 2.8: The full text inclusion/exclusion evaluation process

To ensure a thorough retrieval of relevant studies, the search was revisited. Another search was conducted on Google Scholar with the same final search string. However, similar results from the previous search were retrieved. Again, the same search string was used on Google. This also was unproductive. As a last step, a more generic search string: *(Big data) AND (Probabilistic programming)* was used on Google. After careful consideration of the results by the two researchers based on the initial filtering and abstract evaluation, one (1) paper was selected.

The selected paper was passed onto the second evaluation phase – the full text inclusion evaluation phase (see Figure 2.8). In this phase, the selection process was done in the same way as the abstract evaluation phase. However, the content of the paper was read by both researchers to see if it was relevant to the study based on IC3 criteria.

IC3 The study describes the technique used in the adoption of probabilistic programming in big data processing.

As in the previous phase, all disagreements were carefully deliberated on and resolved. The same selected paper passed this evaluation phase and was given an identifier – S01. The selected study was moved to the quality assessment step.

2.4.2.3 Quality Assessment

This step helps to evaluate the quality of the selected papers, their relevance in answering the research questions, and provides a yardstick for further analysis of the selected studies. According to Kitchenham & Charters (2007), it is important to analyse the quality of the selected primary studies based on the research area. This serves as an additional filtering mechanism to carefully choose more suitable studies from the primary studies that best answer the research questions. Five (5) quality assessment criteria (QAC) were considered to evaluate the quality of the selected primary study. The quality assessment criteria are centred around the following questions:

QAC1 Does the research clearly state an understandable and straightforward statement of aim?

QAC2 Is the study associated with other related research to be fully understood?

QAC3 Does the study presents its data set characteristics (metrics)?

QAC4 Does the research clearly describes its experimental method?

QAC5 Does the study report its performance or validation assessment?

Each researcher read and assessed the selected primary study independently against the QACs. For the selected primary study, if any of the above quality criteria is satisfied, a 'Yes' response is given against that question. In the same way, a 'No' response is given if the paper fails to satisfy a criterion. If the answer to any of the questions could be implicitly inferred, a 'Partly' response is given. Consequently, each response was given a numeric score between 0 and 1 as follows: *Yes = 1, Partly = 0.5 and No = 0*. Again, all disagreements were resolved by discussing until a consensus was reached on a score. Finally, the sum of scores for each response to the QACs was calculated. If the sum is greater than or equal to 2.5, the paper is accepted and used in this systematic review.

2.4.2.4 Data Collection

The process of data collection ensures that data needed to answer the research questions (cf. Section 2.4.1) are extracted from the final primary study. A data extraction spreadsheet was created with the information below:

- Paper title.
- Name of authors.
- Year of publication.
- Source of article (journal or conference).
- Number of pages (if available).

One of the researchers was responsible for data extraction while the other scrutinized the extracted data.

2.4.2.5 Data Synthesis

In this step, a procedure was established to organize, analyse and summarize the result of the quality assessment and data extraction stages. Since only one paper was selected as the primary study, both researchers carried out a discussion session in a face-to-face meeting to respond to the research questions.

2.4.3 Results

In this section, results of the systematic literature review are presented and summarized.

2.4.3.1 Search Results

The total number of initial studies returned after the search on the selected four databases was 293 as shown in Table 2.4. However, after the initial filtering and abstract evaluation, all 293 studies were rejected. Thus, the search phase was revisited with a more generic search string on Google which resulted in the identification of one paper shown in Table 2.5.

Table 2.5: Selected primary study

Identifier	Name of Authors	Publication Year	Paper Title
S01	Zhuoyue Shao, Jialing Pei, Eric Lo, Kenny Q. Zhu, and Chris Liu	2017	InferSpark: Statistical Inference at Scale

2.4.3.2 Quality Evaluation

The selected study was assessed for quality using the formulated questions as presented in Section 2.4.2.3.

Table 2.6: Quality evaluation of selected primary study

Study	QAC 1	QAC 2	QAC 3	QAC 4	QAC 5	Total
S01	1	1	1	1	1	5

Table 2.6 is a summary of the result of the quality assessment phase. This table clearly shows that the selected paper passed the quality assessment criteria with a total score of 5. Thus, the paper was used in this systematic review.

2.4.3.3 Summary of Selected Study

This section presents a summary of the selected primary study used in this review.

Zhao, Pei, Lo, Zhu, & Liu (2017) (S01), presented a probabilistic programming framework on top of Apache Spark called *InferSpark*. According to the authors, this framework claims to aid in the implementation of statistical inference on big data using the distributed main processing power of Apache Spark. They recognized the potentials of probabilistic programming in the development of complex probabilistic models using concise functions or procedures in a programming language. This takes away the responsibility and weight of developing statistical models from the user to the compiler and runtime systems. The authors also noted that although Spark has emerged as an effective solution for large-scale big data processing with its stack of libraries containing statistical models and inference algorithms, users still do not have the liberty to create custom models. InferSpark is an introduction of probabilistic programming on Spark to enable user-defined models and automatic generation of inference algorithms. Zhao et al. (2017) acknowledged the presence of probabilistic programming systems such as Infer.NET, Church and Figaro, but pointed out that the emphasis has been more on the efficiency and effectiveness of the inference algorithm and language. Thus, the scalability of these probabilistic programming frameworks has been ignored.

InferSpark was implemented in two phases. The first was the extension of Scala programming language to support probabilistic programming. The authors chose Scala to benefit from its functional paradigm and because Spark is implemented in Scala. In the second phase, the authors built a compiler and runtime system that compiles InferSpark models into Scala classes using the code generation approach, typically, an input to InferSpark framework consist of a model definition and a normal Scala code. A Bayesian network template is extracted from the model definition and converted into a Scala program at runtime. This converted Scala program is then processed using code generation to produce a Spark program that is executed on Apache Spark.

At the time of publication, InferSpark supports Bayesian network models and implements only variational message passing (VMP) inference algorithm. However, their future work indicated an intention to support other models like the Markov network. InferSpark claims to be the first attempt at introducing the concept of probabilistic programming into the big data domain.

2.4.4 Analysis

This section presents an analysis of the result from the systematic literature review based on the four research questions (cf. Section 2.4.1).

2.4.4.1 SLRQ1: Existing Big Data Solutions using Probabilistic Programming Concept

The review identified one (1) existing framework that contributed to the big data domain using probabilistic programming. As stated in the study selection phase of this review, the identified solution was given an identifier – S01. The authors of S01 named their framework InferSpark.

2.4.4.2 SLRQ2: Comparison of Solutions

There was only one identified solution in this review, thus an answer to the second research question was unnecessary. However, the authors of S01 did compare their solution with existing probabilistic programming frameworks, stating the advantages and limitations. According to the authors, the main limitation of available probabilistic programming systems is the inability to scale out on a distributed computing framework. Again, InferSpark claims to be the only platform that efficiently executes statistical inference on big data using probabilistic programming on Spark. Machine learning libraries like MLlib, Mahout and MADLib can also be used on Spark. However, these machine learning libraries does not support user defined models.

2.4.4.3 SLRQ3: Strength of Evidence in Support of Solution

The authors of S01 presented a performance evaluation of InferSpark. The performance evaluation was based on the construction and execution of statistical inference on three models namely *Latent Dirichlet Allocation (LDA), Sentence-LDA (SLDA),* and *Dirichlet Compound Multinomial LDA (DCMLDA)*. LDA is a type of topic modelling that infers the topics from a collection of documents. SLDA is a model used to discover aspects in online reviews and DCMLDA is another type of topic model that considers sudden explosions of topics. The performance of InferSpark was compared to the performance of MLlib and Infer.NET using the same three models.

The performance evaluation was categorised in three separate levels: Overall performance, Scaling-up, and Scaling-out. The running time of InferSpark in the overall performance level was negligible compared to MLlib and Infer.NET which did not complete the inference task within a week. The authors also demonstrated that InferSpark handles increase in data size and can achieve linear scale-out.

2.4.4.4 SLRQ4: Implications of Findings to this Research

The findings from answering the research questions show that the concept of probabilistic programming is useful to the big data domain. Based on the strength of evidence presented in Section 2.4.4.3, probabilistic programming could go a long way in the easier and efficient development of complex probabilistic models and inference algorithms in the big data domain. This shows relevance of this research. Again, the authors of S01 strongly believe that InferSpark is the first attempt to introduce probabilistic programming concepts in the big data domain; which leaves room for more improvement.

There are some identified limitations of InferSpark which forms the basis for future work. InferSpark only supports the Bayesian network models. Although InferSpark supports Bayesian networks models, at the time of publication, only the VMP inference algorithm was supported.

The authors of S01 did not explicitly emphasize the real-time response from the statistical inference computation in InferSpark. Therefore, a major contribution of this research is a timely response using probabilistic programming and Lambda architecture framework on big data.

2.4.5 Systematic Literature Review Summary

An intensive search and study of probabilistic reasoning big data applications that used the concept of probabilistic programming was presented in this systematic literature review.

This review identified one (1) available framework that combined the concept of probabilistic programming and Apache Spark. The framework called InferSpark was developed on the Scala programming language with the main aim of introducing probabilistic programming into the big data domain. InferSpark attempts to solve the scalability limitation of available probabilistic programming systems as well as the inability of users to design custom models on Spark.

Although InferSpark performed well compared to other machine learning libraries on distributed computing frameworks and Infer.NET (an example of a probabilistic programming system), it only implemented a specific inference algorithm in the Bayesian network model called the Variational Messaging Passing (VMP). Again, the authors did not clearly emphasize the real-time aspect of big data. Thus, this research seeks to develop a probabilistic reasoning big data application using probabilistic programming and Lambda architecture to give real-time response to queries.

2.4.6 Systematic Literature Review Limitations

The field of software engineering lacks well established and generally accepted procedures and conduct for performing a systematic literature review unlike other disciplines such as medicine (Staples & Niazi, 2007). As supported and reported by

Brereton, Kitchenham, Budgen, Turner, & Khalil (2007) in their paper titled *Lessons from applying the systematic literature review process within the software engineering domain*, available software engineering online databases or search engines does not assist systematic reviews in contrast to medical sciences, and the standard of abstracts in software engineering papers are not always suitable to determine the importance of a study. Thus, this presents difficulty in carrying out a systematic review in software engineering.

A major challenge in carrying out a systematic literature review is the selection of relevant primary studies (Kitchenham et al., 2010). In this case, four (4) online databases were used as the source for primary studies. It is possible that some papers may have been missed because of the search terms and their combination. An example is the “Big data processing” search category; it is possible that some papers that may be relevant did not include the search term in their abstract and conclusion.

This review also excluded technical reports, graduate theses and newsletters. Another possible limitation in this review is that one researcher was responsible for extraction and the other was tasked with reviewing as recommended by Budgen & Brereton (2006), this may attract bias in both stages of extraction and review.

2.5 Chapter Summary

This chapter introduced the knowledge, theories and background related to this research.

Firstly, a description of big data was presented, with its concept, definition, history and processing paradigms. Then an introduction to the concept of Lambda architecture as a unifying framework for both batch and stream processing. The next section presented a discussion of probabilistic reasoning, its application and relevance to big data especially in handling data uncertainty. Then the research area called probabilistic programming was presented in detail, with emphasis on how it enables easier development of probabilistic models and automatic generation of inference algorithm. Finally, a systematic review with the goal of identifying big data applications that leverage the idea of probabilistic programming was presented to highlight the relevance of this study.

CHAPTER THREE
METHODOLOGY

A brief description of the research methodology to be used in this study was introduced in Section 1.7 of the first chapter. This chapter presents a detailed narration of the research paradigm used in this thesis.

The purpose of this chapter is not to elaborate on the foundations of research methodologies and philosophical rationale. However, this chapter discusses a suitable choice of research methodology for studies in the technical discipline. The choice of research paradigm and methodology is justified using the research pyramid presented by Jonker & Pennink (2009).

3.1 Introduction

The process associated with scholarly and rigorous discovery of beliefs, theories and opinions to improve, broaden or authenticate knowledge and understanding of the world around us is known as *research* (Ryan et al., 2002; Geerts, 2011). According to Cambridge English Dictionary (2011) and Oxford English Dictionaries (2018), research is a thorough exploration of a particular subject to uncover new information.

Research should be conducted in a defined and suitable method to be relevant, important and contribute to the body of knowledge. A research method, sometimes called research strategy (Järvelin & Vakkary, 1990), research framework (White & Marsh, 2006), research methodology (Hildreth & Aytac, 2007), or research design (Luo & McKinney, 2015) is a well ordered, standardized way or strategy to carrying out research (Crotty, 1998).

Looking at the primary aim of this research as outlined in Section 1.4 of Chapter one, this study seeks to develop a scalable tool that will efficiently process big data in real-time and produce a probabilistic score. Simply put, this research will produce an artefact. Thus, the design science research will be used in this study as it is relevant to studies in technical fields such as architecture, computer science, information technology and engineering (Simon, 1997; Weber, 2010).

3.2 Research Pyramid

In a bid to help researchers structure the actions of their study based on their research questions, Jonker & Pennink (2009) introduced the research pyramid

(Figure 3.1). According to the authors, the pyramid serves as a direction on how researchers are to outline a suitable and justifiable research methodology.

The research pyramid is divided into four stages namely *Research Paradigm, Research Methodology, Research Methods* and *Research Techniques.* The stages should be viewed as interconnected actions starting from an abstract (top) level to a more practical (bottom) action.

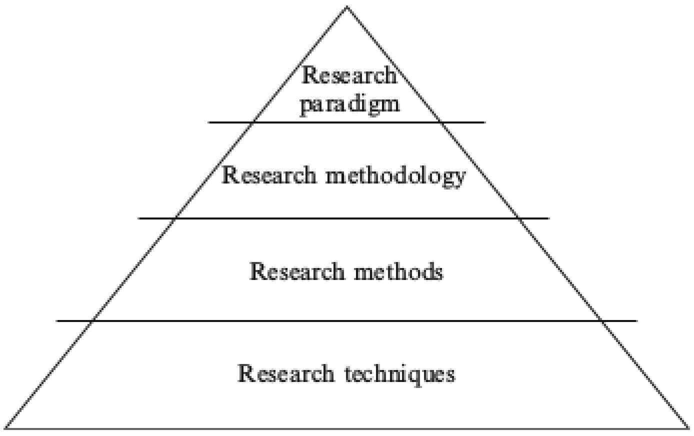


Figure 3.1: Research Pyramid

(Jonker & Pennink, 2009: 23)

In each of these phases, the research must make a suitable choice based on the nature of the research question(s). A research methodology is chosen based on a selected paradigm that best handles the specific research. Furthermore, an appropriate method associated with the chosen methodology is selected and implemented using fitting research techniques.

3.2.1 Research Paradigm

Every researcher has a different opinion on the constituents of truth and knowledge. These different views direct different thinking, beliefs and assumptions about the world around us. A paradigm is referred to by social scientists as the way different professions view the world around them (Schwandt, 2001). Kuhn (1970) describe paradigm to represent a common set of beliefs and assumptions shared across a specific subject on how problems should be solved. Guba (1990) presented a more generic definition of paradigm: a set of firmly held opinions that steers actions and characterized by what reality is – *ontology*, how reality or knowledge is known – *epistemology*, and how knowledge is found – *methodology* (Crotty, 1998).

A number of research paradigms are described in literature such as *positivism, constructivism, interpretivism, emancipatory, critical* and *pragmatism* (Mackenzie & Knipe, 2006). The most common are positivism, constructivism and pragmatism.

In positivism, studies are carried out using the scientific approach. This paradigm assumes that there is a single source of truth which is justifiable by scientific methods. Thus, quantitative methods are typically used to measure knowledge. On the other hand, constructivism is associated with the belief that knowledge needs to be interpreted using qualitative methods leading to multiple sources of truth. Pragmatism however, is not bound to a philosophy. Pragmatists view reality as continuously changing thus requiring the most suitable method that leads to knowledge (Guba, 1990; Healy & Perry, 2000; Schwandt, 2001; Walliman, 2001; Mackenzie & Knipe, 2006; Creswell, 2009; Patel, 2015).

Choosing a research paradigm is usually the first step that sets the foundation for subsequent selection of methodology, methods and techniques. This research is action-based and follows the pragmatic research paradigm which is relevant in studies done in technical disciplines.

3.2.2 Research Methodology

The second level of the research pyramid is concerned with the specific way to conduct research based on the selected view of reality. As mentioned in Section 3.1, research methodology describes the global procedures associated with the research process without specifying individual actions. In this way, methodology acts like a main outline of the research approach (Jonker & Pennink, 2009).

Table 3.1: Research paradigms with associated methodology and methods
(Adopted from Creswell, 2009; Mackenzie & Knipe, 2006; Patel, 2015)

Paradigm	Methodology	Methods
Positivism	Experimental research. Survey research. Deterministic. Normative	Mostly quantitative such as measurement, sampling, interviews, focus groups, etc.
Constructivism	Ethnography. Discourse analysis. Naturalistic.	Mostly qualitative and could include case study, narrative, etc.
Pragmatism	Design-based research. Action research. Participatory research. Mixed models.	Mixture of qualitative and quantitative methods, etc.

Methodologies associated with the pragmatic paradigm are usually design-based, action research (see Table 3.1). This research seeks to create an artefact. The Design Science Research Methodology (DSRM) is relevant in studies that are inclined to the actualization of an artefact. Thus, this research uses the DSRM and follows the design process outlined by Peffers et al. (2007) (cf. Section 3.4). This study also adopts the guidelines as proposed by A. R. Hevner et al. (2004).

3.2.3 Research Methods

The research methodology is a generic specification on how to conduct a research (cf. Section 3.1 and 3.2.2). A more specific procedure, sequence of activities or technique in the research methodology is provided in the research methods (Crotty, 1998; Mackenzie & Knipe, 2006; Jonker & Pennink, 2009). These specific steps are presented in an order describing how the research is carried out (see Table 3.1).

A. R. Hevner et al. (2004) recommended appropriate research methods for DSRM. This research adopts the suggestions of A. R. Hevner et al. (2004) and is described in detail later in this chapter.

3.2.4 Research Techniques

Research techniques describes a more concrete and practical execution of the research methods. This is the fourth level of the research pyramid. Research techniques provides detailed instructions (including instruments and tools) used in the selected research method and "…can be understood as concrete instructions for acting that have an explicit, compelling and prescribing character." (Jonker & Pennink, 2009: 34).

The techniques used in this research is adopted from A. R. Hevner et al. (2004) and is further discussed in details in Chapter 5.

3.3 Design Science Research

Design Science Research (DSR) is used to solve real-world problems by developing novel solutions (Walls et al., 1992). According to Glass (1999) and Simon (1997), DSR originated from the engineering discipline and its relevance in information technology/systems research has been acknowledged. Benbasat & Zmud (1999) pointed out that the appropriateness of DSR in information systems research can be tied to its relevance in design. However, Gregor & Hevner (2013) and Peffers et al. (2007) stated that its adoption is relatively slow in information technology research.

In information systems, DSR deals with the development of artefacts that involves people and technology (Gregor & Hevner, 2013). Similarly, Strode & Chard (2014) defined DSR as a research paradigm in information technology which is based on the pragmatism approach and mostly used by software engineering researchers to contribute to the body of knowledge through exhaustive development and evaluation of an artefact. Comparing research in information technology and natural sciences, March & Smith (1995) indicated that design science is closely associated with research in technology-oriented fields and produces utilities that contribute to human purposes. According to Vaishnavi, Kuechler, & Petter (2004), DSR is acquiring or improving knowledge through the construction of an artefact. In other words, DSR involves research using the process of invention (design) as a research method (Myers & Venable, 2014).

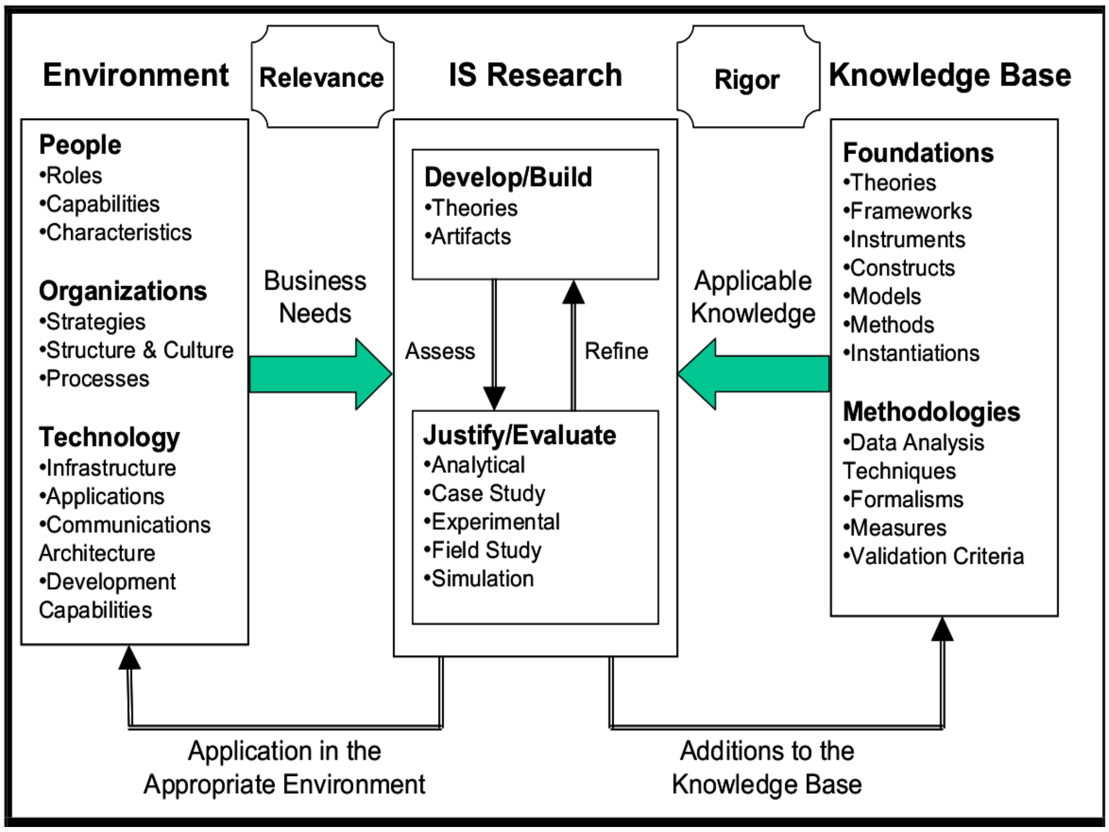


Figure 3.2: Information Systems DSR Framework

(Hevner et al., 2004: 80)

Walls et al. (1992) describes design as both a series of actions (process) and an invention. The design process leads to the actualization of a novel product which is evaluated. The assessment of the artefact enables improvement in the value of the artefact as well as the design process (Hevner et al., 2004). This build and evaluate procedure is listed as the two iterative design processes associated with DSR (March & Smith, 1995; Markus et al., 2002; Peffers et al., 2007). March & Smith (1995) also outlined four design outputs of DSR namely *constructs* (corresponding to techniques, representations and conceptual theories), *models* (representing abstractions), *methods* (procedures, practices or routines), and *instantiations* (prototype or concrete implementations). Design science artefacts should demonstrate relevance by providing a solution to a problem either by introducing a novel solution or improving an existing solution (Weber, 2010; Geerts, 2011).

Building upon the foundation of March & Smith (1995), A. R. Hevner et al. (2004), proposed a conceptual framework (see Figure 3.2) for research in information systems to assist in proper comprehension, implementation and evaluation of DSR. This framework has been adopted by most researchers (Choi et al., 2010; Rodríguez et al., 2014; Strode & Chard, 2014). The framework shows the influence of environment and knowledge on research in information systems as well as the relationship between them. The combination of environment and knowledge base provides relevance and rigor to studies in information systems.

3.3.1 Business Needs

The framework in Figure 3.2 clearly shows business needs as one of the influencing ingredients that contribute to research in information systems. The source of business needs is the specific environment as relates to the information systems research. The environment which consists usually of technology and people within an organisation makes up the problem space. As mentioned in Section 3.3, relevance is one of the defining characteristics of a design artefact. The problem space provides relevance for an information systems research (Hevner et al., 2004).

A desired goal of this research is to aid in the process of decision-making by providing real-time response to big data analysis especially in uncertain circumstances (cf. Section 1.5). Decision-making is key in policy formulation and thus provides the specific problem space for this research. All three components of environment (people, organization and technology) influences and gives rise to the business needs of a real-time probabilistic reasoning system.

3.3.2 Applicable Knowledge

The other important component of research in information systems is the knowledge base. According to Hevner et al. (2004), foundations and methodology form the basis from and through which information systems studies are successfully achieved. Prior literature containing foundational theories and instantiations provide the raw material that helps in the build phase of DSR while methodologies provide the tools for the evaluation phase. Thus, the effective application of foundation and methodology in information systems research ensures rigor. This satisfies the ‘rigor’ characteristics of a design artefact.

In this research, theoretical foundations from available big data tools and technologies as well as literature on big data, big data analysis, probabilistic reasoning, probabilistic programming, machine learning, and lambda architecture were studied. Furthermore, appropriate methodology as proposed by Peffers et al. (2007) is used in this research and evaluation is done against the guidelines as outlined by A. R. Hevner et al. (2004).

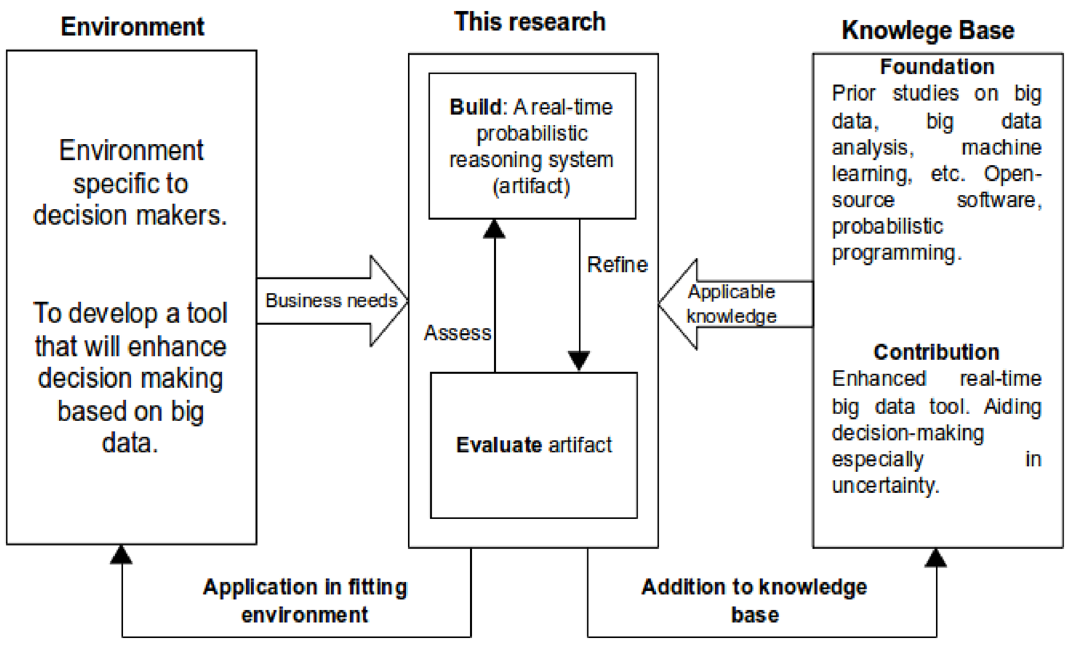

Environment
Environment specific to decision makers.
To develop a tool that will enhance decision making based on big data.
Business needs
This research
Build: A real-time probabilistic reasoning system (artifact)
Refine
Assess
Evaluate artifact
Applicable knowledge
Knowlege Base
Foundation
Prior studies on big data, big data analysis, machine learning, etc. Open-source software, probabilistic programming.
Contribution
Enhanced real-time big data tool. Aiding decision-making especially in uncertainty.
Application in fitting environment
Addition to knowledge base

Figure 3.3: An adaptation of DSR framework

(Hevner et al., 2004: 80)

Figure 3.3 is an adaptation of the DSR framework illustrating the specific environment, knowledge base and research contributions.

3.4 Design Science Research Methodology

In recent times, researchers have successfully and explicitly shown the relevance of design science in information systems research (Nunamaker et al., 1990; Walls et al., 1992; March & Smith, 1995). However, as Peffers et al. (2007) pointed out, these studies did not clearly describe a common methodology for design science research in information systems.

To effect a useful contribution to the body of knowledge, Jonker & Pennink (2009) included research methodology as one of the stages of a research (cf. Section 3.2). Peffers et al. (2007) argued that a common research methodology constitutes a mental template for the comprehension and evaluation of a research. Thus, building on past studies, Peffers et al. (2007) recommended a design science research methodology to help in the construction and arrangement of design science research in information systems. According to Peffers et al. (2007), a design science research methodology should include principles of design science, rules and a process model (see Figure 3.4). The proposed DSRM process claims to serve as a strategy and mental model for researchers using design science research.

This research acknowledges there are other methodologies proposed in literature (Vaishnavi et al., 2004; Venable, 2006; Baskerville et al., 2009; Iivari & Venable, 2009; Bilandzic & Venable, 2011; Sein et al., 2011). However, this research is based on the DSRM process model in Figure 3.4. This choice is justified later in this chapter against the guidelines presented by A. R. Hevner et al. (2004).

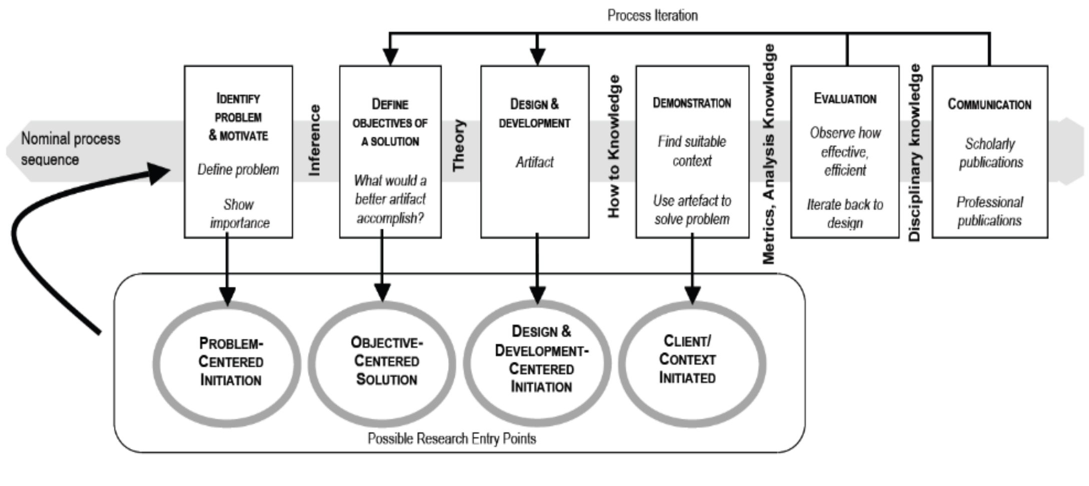

Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development-Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

Figure 3.4: DSRM Process Model

(Peffers et al., 2007: 54)

The DSRM process model is made up of six actions as illustrated in Figure 3.4. The process model was designed to be sequential, however, Peffers et al. (2007) emphasized four different entry points to the process model based on the actual research or study. The first entry point is the *Problem-centred initiation* resulting from an identified problem or recommended future study in research papers. *Objective-centred solution* is the second point of entry tied to the second activity and could be because of an industry or research need. The third entry point is the *Design & Development-centred initiation*. The third entry point starts with activity 3 and could be stimulated from a known but not implemented artefact. The *Client/Context initiated* entry point starts with the fourth activity and could be caused by a real project.

The study described in this thesis lies in the *Problem-centred initiation* entry point and starts from activity 1.

3.4.1 Activity 1 – Identify Problem and Motivate

In this activity the specific problem that the research seeks to tackle should be clearly established along with the justification for a solution (Peffers et al., 2007).

Decision-makers are often required to make rapid decisions based on current events and past experiences. Current events represent real-time data while past experiences denote static or historic data, both of which constitutes big data. Research in big data processing has led to the development of innovative technologies to process big real-time data and static data. However, these technologies exist independently to solve specific big data (batch and streaming) problems (cf. Section 1.2, 1.3 and 2.1.4).

This research focuses on real-time reasoning with big data. By the effective combination of the available open-source big data tools and technologies based on Lambda architecture as proposed by Marz & Warren (2015), the focus of this research is to provide a real-time probabilistic reasoning solution built using probabilistic programming that will aid in decision-making especially in terms of uncertainty.

3.4.2 Activity 2 – Define Objectives of a Solution

The second activity according to the DSRM process as proposed by Peffers et al. (2007) involves defining the objectives of a solution to the identified problem in activity 1. According to Peffers et al. (2007), the objectives of a solution denotes what is practical and attainable and can be either quantitative or qualitative.

In chapter 1, a brief explanation of three objectives for this research was presented (cf. Section 1.5). These objectives were also substantiated against literature (cf. Section 1.3 and 2.1). Furthermore, detailed objectives about the design of the artefact are presented in chapter 4 (cf. Section 4.2).

3.4.3 Activity 3 – Design & Development

Activity 3 consists of the actual design and development of the artefact which can be in the form of constructs, models, methods, instantiations or a combination of any

(March & Smith, 1995; Hevner et al., 2004). Peffers et al. (2007) pointed out that knowledge of theory relevant to problem space is required to translate the defined objectives into actual features and functionalities, thus creating a design research artefact with embedded research contributions.

Chapter 4 describes and documents the design and development stage of this design research artefact based on the identified objectives in activity 3.

3.4.4 Activity 4 – Demonstration

The efficacy of the artefact is required to be demonstrated in one or more occurrences of the problem domain or space. According to Peffers et al. (2007), demonstration could be a case study, proof, experimental or simulation.

Details of the demonstration are presented in chapter 5. In this case, demonstration is presented as a case study.

3.4.5 Activity 5 – Evaluation

This activity involves monitoring and assessing how well the artefact tackles the problem in accordance to the identified objectives defined in activity 3. Peffers et al. (2007) stated that comprehension of applicable metrics and analysis procedures is required to carry out this activity. March & Smith (1995) and Hevner et al. (2004) acknowledged that evaluation is crucial as it shows the usefulness and benefits of the design artefact.

Hevner et al. (2004) listed five categories of evaluation techniques namely observational, experimental, descriptive, testing and analytical. This research will use the experimental evaluation method (cf. Section 3.5.3) to assess the design artefact. Experimentation will be through simulation with real data. Details of the evaluation are described in chapter 5.

3.4.6 Activity 6 – Communication

Peffers et al. (2007) stated the importance of communicating the research problem, developed artefact and design rigor to other researchers and appropriate audiences. This point is also supported by A. R. Hevner et al. (2004). Gregor & Hevner (2013) and Strode & Chard (2014) proposed a structure for reporting research done using design science (see Table 3.2).

This thesis forms a major means of communicating this research and adopts the publication schema as presented by Gregor & Hevner (2013) and Strode & Chard (2014).

Table 3.2: DSR Publication Structure

(Adopted from Gregor & Hevner, 2013: A5; Strode & Chard, 2014: 244)

Section	**Contents**	**Thesis Chapter**
Introduction	*Problem definition, problem significance/motivation, introduction to key concepts, research questions/objectives, scope of study, overview of methods and finding, theoretical and practical significance, structure of remainder of paper.* For DSR, the contents are similar, but the problem definition and research objectives should specify the **goals** that are required of the artefact to be developed.	Chapter 1
Literature Review	*Prior work that is relevant to the study, including theories, empirical research studies and findings/reports from practice.* For DSR work, the prior literature surveyed should include any prior design theory/knowledge relating to the problem to be addressed, including artefacts that have already been developed to solve similar problems.	Chapter 2
Method	*The research approach that was employed.* For DSR work, the specific DSR approach adopted should be explained, with reference to existing authorities.	Chapter 3
Artefact Description (*Main section of report*)	*A concise description of the artefact at the appropriate level of abstraction to make a new contribution to the knowledge base.* This section (or sections) should occupy the major part of the paper. The format is likely to be variable but should at least contain the description of the designed artefact and, perhaps the design search process.	Chapter 4
Evaluation	*Evidence that the artefact is useful.* The artefact is evaluated to demonstrate its worth with evidence addressing criteria such as validity, utility, quality and efficacy.	Chapter 5
Discussion	*Interpretation of the results: what the results mean and how they relate back to the objectives stated in the Introduction section. Can include: summary of what was learned, comparison with prior work, limitations, theoretical significance, practical significance, and areas requiring further work.* Research contributions are highlighted and the broad implication of the paper's result to research and practice are discussed.	Chapter 7
Conclusions	*Concluding paragraphs that restate the important findings of the work.* Restates the main ideas in the contribution and why they are important.	Chapter 7

3.5 Design Science Research Guidelines

A. R. Hevner et al. (2004) identified design science as a problem-solving technique where the knowledge of a design problem and its solution are obtained in the development and use of an artefact. They suggested seven guidelines for conducting and evaluating research processes in design science. This section describes how this study intends to follow the rule for an effective design science research.

3.5.1 Guideline 1 – Design as an Artefact

The first guideline requires the output of a design science research to be an artefact. An IT artefact is defined to be one of a combination of construct, model, methods or instantiations (cf. Section 3.3). Artefacts developed in research using design science are usually not exhaustive and comprehensive information systems. However, design science artefacts show innovations that form the basis of knowledge through design (Hevner et al., 2004; Gregor & Hevner, 2013).

The output of this research is an instantiation (artefact) – implementation of a real-time probabilistic reasoning system using Lambda architecture.

3.5.2 Guideline 2 – Problem Relevance

The problem space or specific environment determines the relevance of a DSR endeavour. A research using design science must solve important problem(s) faced by people, organizations and technology. The problem relevance provides both the requirements and the evaluation standard for the research and research outputs respectively (Simon, 1997; Hevner et al., 2004; Hevner & Chatterjee, 2010).

The background to this study and problem statement (cf. Section 1.2 and 1.3) as well as chapter 2 provides enough insight to the problem space and relevance of this research.

3.5.3 Guideline 3 – Design Evaluation

This is important to any DSR process (Simon, 1997). The artefact of a design science research should be measured against a well-established metrics based on established requirements using *observational, analytical, experimental, testing* and/or *descriptive* evaluation methods to ascertain if it works and how well it works. Evaluation provides feedback on the quality and efficacy of the product (Hevner et al., 2004; Hevner & Chatterjee, 2010).

In the case of this research, the artefact is measured against the research questions and research goals (cf. Section 1.5 and 1.4). This research will also use the experimental evaluation method to evaluate the design artefact. Experimental evaluation will be conducted using simulation with artificial data.

3.5.4 Guideline 4 – Research Contribution

Research in design science must provide comprehensible contributions to knowledge base. This contributions may be in the form of a design artefact, foundation theories and/or methodologies (Hevner et al., 2004).

This research contributes in the form of an artefact that applied prior knowledge in new and innovative ways to solve the research problems highlighted in Section 1.3. Details of contributions are described more in chapters 4 and 7.

3.5.5 Guideline 5 – Research Rigor

Rigor pertains to the soundness of the methods in the research process of building and evaluating of the design product. Design science research requires "…the application of rigorous methods in both the construction and evaluation of the designed artefact." (Hevner et al., 2004: 87).

Table 3.3: DSR Guidelines

(Adopted from Hevner et al., 2004: 83; Hevner & Chatterjee, 2010: 12)

Guideline	Description	Mapping to this research
Design as an Artefact	Design science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation.	Real-time probabilistic reasoning system using Lambda architecture.
Problem Relevance	The objective of design science research is to develop technology-based solutions to important and relevant business problems.	Improve decision-making. Provide inexpensive automated big data solution.
Design Evaluation	The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.	Experimental Evaluation consisting of simulation and testbed.
Research Contribution	Effective design science research must provide clear and verifiable contributions in the design artefact, design foundations, and/or design methodology.	Artefact that uses Lambda architecture on inexpensive commodity hardware to enhance decision-making.
Research Rigor	Design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.	Methodology based on prior authority.
Design as a Search Process	The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.	Literature review in the context of big data, big data processing, probabilistic reasoning, Lambda architecture. Open-source big data tools and technologies.
Communication of Research	Design science research must be presented effectively to both technology-oriented and management-oriented audiences.	Thesis & publications.

This research will use foundational theories in big data, big data processing, machine learning, probabilistic programming, and current big data techniques and technologies as its foundation (cf. Chapter 2). The design and development of the artefact in this research is described in Chapter 4 and as mentioned in Section 3.5.3, the experimental evaluation method is used in the evaluation stage.

3.5.6 Guideline 6 – Design as a Search Process

Design is an iterative process in the context of a problem environment, to reach an effective solution (Hevner et al., 2004; Gregor & Hevner, 2013).

In chapter 2, this study presented a rigorous examination of literature in the problem domain to present objectives to accomplish an effective solution. The search process continues in chapter 4, illustrating the design and development of the artefact based on literature. Chapter 5 provides evidence that the artefact is useful in the specified problem domain.

3.5.7 Guideline 7 – Communication of Research

Design science research must be effectively communicated to both technology and management-oriented audiences. This communication should include the description of the construction of the artefact, and details for organisation to buy, build and use the artefact (Hevner et al., 2004).

This study satisfies this guideline with a written thesis as the main piece of communication targeted towards the academic audience. However, a few peer-reviewed conference papers and learning materials are also used to communicate this research.

Table 3.3 shows a summary of the mapping of this research to the guidelines discussed above.

3.6 Chapter Summary

This chapter presented a research methodology for this study using the research pyramid presented by Jonker & Pennink (2009) as a high level framework.

This research chose the design science research which is based on the pragmatic research paradigm. Out of the methodologies in design science available in literature, this research used the methodology as proposed by Peffers et al. (2007). This methodology is examined against the guidelines outlined by A. R. Hevner et al. (2004) to present a complete research methodology used to tackle the research questions.

CHAPTER FOUR
DESIGN OF A REAL-TIME PROBABILISTIC REASONING SYSTEM USING LAMBDA ARCHITECTURE

In chapter 1, an introduction to this research was presented with a brief description of the context of the problem and clearly defined aim and objectives. Chapter 2 provided the background knowledge relevant to this study as well as a systematic literature review of big data applications that used the probabilistic programming concept. Chapter 3 provided a detailed description of a suitable research methodology for this study.

This chapter builds upon the findings from chapter 2 and the chosen research methodology discussed in Chapter 3 to present the design of an artefact with respect to the second goal of this research outlined in chapter 1.

The first section of this chapter gives a short introduction to the background and aim for the real-time probabilistic reasoning system using Lambda architecture (RT-PRLA). Section 4.2 provides a list of requirements for the design of the RT-PRLA system. In section 4.3, a generic system overview is presented, detailing the rationale behind the design decision. Section 4.3 also presents the various components of the system and how they are combined using Lambda architecture to form a real-time probabilistic reasoning system. Section 4.4 provides a descriptive detail of the design of the components discussed in Section 4.3. Section 4.5 summarizes the entire chapter.

4.1 Introduction

The rationale behind the development of a real-time probabilistic reasoning system was influenced by several factors as mentioned in chapter 1, however, the most striking reason is the need for a big data application that performs probabilistic inference on both real-time and batch data to output real-time response that will enable decision making in uncertain situations. This need was highlighted in section 2.4 of chapter 2. Another motivation is to build upon existing big data infrastructure, tools and technologies described in chapter 2.

This chapter restates the design objectives of the proposed artefact and describes the design process. This is in accordance to the second and third activities in the design science research methodology model as proposed by Peffers et al. (2007). Activity 2 is all about defining objectives for a solution while activity 3 is concerned with the actual design and creation of the solution. According to Peffers et al. (2007), knowledge required to carry out activity 2 includes having an understanding about the current state and solutions. Activity 3 requires knowledge of relevant fields and technology. In this research, the relevant areas are big data processing, Lambda architecture and probabilistic programming.

In summary, the goal of this chapter is to present the design of an artefact according to the aim of this research to satisfy the following:

- Provide real-time response to big data processing.
- Use the concept of Lambda architecture to effectively combine batch and stream processing.
- Use probabilistic programming for easier development of probabilistic models and automated inference algorithms.

4.2 Design Objectives/Requirements

In the early phase of this thesis, the purpose of this research was presented. The purpose is to support decision-makers in the process of decision making especially in uncertain circumstances using automated, real-time probabilistic reasoning system. Thus, a list of requirements was identified (cf. Section 1.5).

Probabilistic reasoning systems already exist to support decision-making. However, after conducting a background and systematic literature review, new concepts, trends and recommendations that may better handle contemporary demands were identified and the identified list of objectives was reviewed, re-established and formalized as DR1 to DR6.

The list of objectives (DR1 to DR6) is motivated by technological innovation and represents high level requirements, thus they do not specify any techniques to use in their actualization. The specific methods used to achieve these objectives are explained in the design presented in Section 4.4.

DR1 – Data Integration: The system *must* efficiently handle data streams from multiple sources as well as integrate both real-time and batch (historical) data.

DR2 – Data Ingestion and Overflow: The system *must* adequately manage data ingestion and back pressure.

DR3 – Real-time Computation: The system *must* perform computation and provide response in real-time in the form of a probabilistic score, considering limited computational resources.

DR4 – Learning Capability: The system *must* provide learning features based on probabilistic reasoning.

DR5 – Data Value: The system *should* efficiently compress and filter out irrelevant data yet not affecting the potential value of data.
DR6 – Generic Context: The system *should* be generic; used in different context.

The requirements are grouped according to priority level. The high priority requirements *must* be fulfilled and the requirements that *should* be addressed are in the low priority level.

4.3 System Overview

This section describes a detailed design of the artefact in addition to the explanation of the design choices that lead to the proposed solution.

A major distinct motivation for a probabilistic reasoning system is to support decision-making in real-time especially in uncertain circumstances (Gonzalez, 2012; Pfeffer, 2016). This is a vital definite feature and remains the comprehensive purpose of this research. The solution proposed in this thesis is to use Lambda architecture to improve low latency in big data processing and probabilistic programming to aid in easier development of complex models.

Available big data solutions are combined as explained by Marz & Warren (2015) to achieve Lambda architecture. The principal purpose of the Lambda architecture is to build a three-layer generic big data solution, with each layer satisfying a property of big data (3Vs: Volume, Variety and Velocity). The three layers are Batch, Speed and Serving layers (cf. Section 2.2).

The Batch layer is the base and core of the Lambda architecture and tackles the Volume property of big data. This layer houses all data and computation is done periodically on the data to produce a *batch view*. Thus, the batch view is indexed and saved so it can be retrieved quickly with random reads, without having to read all the data when a query is submitted. In summary, the batch layer must be able to store an immutable, constantly growing dataset, and produces precomputed views from the dataset.

On top of the batch layer is the serving layer. The serving layer holds the precomputed batch views produced by the batch layer. This layer is made up of a distributed database that supports random reads and batch updates from the batch layer. It is important to note that the serving layer should not support random writes.

Both the serving and batch layer deals with the volume and variety property of big data. The only trade-off is that the precomputation done in the batch layer has high latency. Thus, new data captured while computation is on-going in the batch layer is not included in the batch view. This means that the serving layer is updated with batch views that do not contain data that came into the master dataset while precomputation was running. The speed layer compensates for this ‘lost’ data.

In the speed layer, computation is performed on ‘new’ or recent data. This ensures that new data is represented when the value of a query is needed. The speed layer is like the batch layer as it also produces a view (*real-time view*) based on computation on recent data it receives. The real-time views are stored on a database that supports random writes and random reads.

However, there are two major differences between the batch and speed layers. Batch layer performs computation on all data in the master dataset at once while the speed layer performs computation on recent data. Again, to achieve low latency, instead of the re-computation done in the batch layer, the speed layer uses incremental computation, this means that it performs computation as it receives recent data and updates the real-time views. Thus, the real-time views get updated very quickly after new data enters the system. The definition of ‘quickly’ is relative to the application, however it is usually between a few milliseconds to a few minutes (Marz & Warren, 2015). A point to note is that once the recent data is captured and processed in the batch layer to produce batch views, the corresponding real-time views are discarded.

Computations in the batch and speed layers are usually implemented using available big data technologies like Apache Hadoop for the batch layer and Apache Storm or Apache Spark for the speed layer. However, the distinct feature of this research is to employ the idea of probabilistic reasoning using probabilistic programming in the implementation of computations done in the batch and speed layers of the RT-PRLA system (see Section 4.4.3).

Results from computations on the batch and speed layers are stored as batch and real-time views respectively and are used with inference algorithms to perform probabilistic reasoning in response to a query.

4.3.1 Components of Real-time Probabilistic Reasoning System using Lambda Architecture

The real-time probabilistic reasoning system is divided into three components: the server, feeder and storage component. These components have the three layers of the Lambda architecture included in them. A brief description of these components is provided in this section.

4.3.1.1 The Feeder Component

The feeder component acts as the one-way input-only door keeper to the RT-PRLA system. Data flows into the RT-PRLA system through the feeder component. The feeder component is responsible for pre-processing the data to reduce redundancy and filter irrelevant data using a compression and filtering mechanism. This takes care of the design requirement DR5. The feeder component is also responsible for integrating data streams from multiple sources entering the RT-PRLA system. This takes care of the design requirement DR1. The feeder component also manages data inflow into the reasoning system to prevent data inflow from multiple sources from overwhelming the RT-PRLA system. This addresses the design requirement DR3.

4.3.1.2 The Storage Component

The role of the storage component is to house data needed by the reasoning system. In the RT-PRLA system, the storage component is made up of four (4) standalone databases, two of which are associated with the batch module of the server component and the other two are associated with the real-time module in the server component. For convenience, these four databases are named *master* database, batch-view database, pseudo-master database, and real-time-view database (see Figure 4.1).

The master database holds the immutable, constantly growing master dataset and supports batch reads and random writes. The master database is part of the implementation of the batch layer of the Lambda architecture. Extreme care must be taken to prevent the master database from any kind of corruption because it acts as the only source of truth to the system (Marz & Warren, 2015).

The other database associated with the batch module of the server component, called the batch-view database serves as part of the serving layer of the Lambda architecture. It holds the result of the pre-computation done in the batch module. It supports batch updates and random reads.

The other two storage (pseudo-master and real-time view) databases are used by the real-time module of the server component. They are used to support real-time computation on new data. The pseudo-master database is used to incrementally store new data as it arrives into the system. This supports random writes. The real-time-view databases hold the result of the incremental computation on the recent data known as real-time views and supports random reads. Both the pseudo-master database and the real-time-view database form part of the speed layer of the Lambda architecture.

4.3.1.3 The Server Component

The server component is responsible for all computation on data. The server component is central to the design concept of the RT-PRLA system. It retrieves data from the immutable data store (master database) as well as from the pseudo-master database and performs computation to produce the batch views and real-time views respectively. Thus, the server component is divided into two modules, on one hand is the *batch* module, and on the other side is the *real-time* module.

The Batch Module

The batch module periodically retrieves the entire data from the master database, performs computation to produce the batch view and subsequently stores the batch view in the batch-view database which corresponds to the serving layer of the Lambda architecture.

The batch module is also part of the batch layer of the Lambda architecture. According to Marz & Warren (2015), the batch layer of Lambda architecture should be able to store immutable master dataset, and perform computation on the dataset. Result from this computation should also be stored as views on the serving layer. However, in this design, the task of storing immutable dataset has been moved to the feeder component (cf. Section 4.3.1.1). This design choice was to remove the initial database write process from the write-read-write process of the batch layer. In other words, instead of having the batch module write to an immutable database, read from the same database to compute arbitrary function on that dataset, and then write the output to the serving layer, it only reads from the immutable dataset that has already been prepopulated by the feeder component. Thus, the batch module only has a read-write process.

The Real-Time Module

The real-time module is responsible for performing incremental computation on real-time data, that is data that came in while the precomputation on the batch module was running. The speed layer of the Lambda architecture is implemented in the real-time module. The real-time module is different from the batch module in that it does not perform a re-computation of all new data; instead it performs incremental computation on recent data as they arrive. This is to ensure low latency updates. As soon as the batch module processes the recent data, the corresponding real-time views are discarded.

4.3.2 Real-time Probabilistic Reasoning Process

This section explains the way the RT-PRLA system works for a better understanding. The various components mentioned in the preceding section are put together to elucidate the general process in the real-time probabilistic reasoning process.

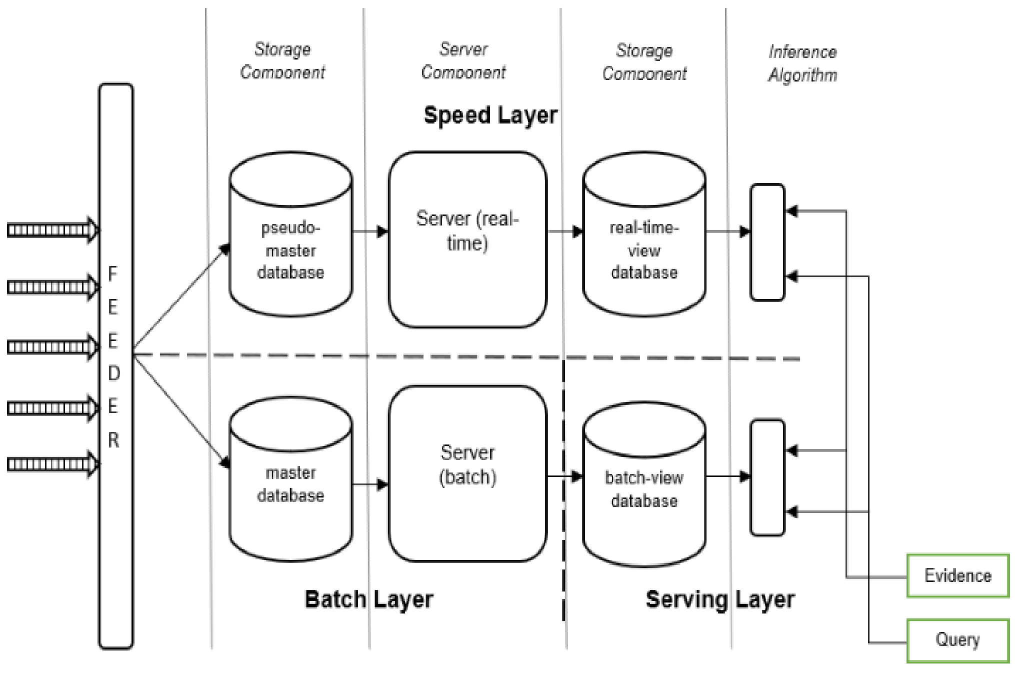


Figure 4.1: Reasoning Framework

Data flows into the RT-PRLA system through a single data pipeline. This data pipeline is used to aggregate data streams from different sources and filter the data to eliminate redundancy. The data pipeline now emits the pre-processed data into the system in two different branches. One branch goes to the master database for batch computation, and the other goes to the pseudo-master dataset for real-time processing.

The RT-PRLA system has a server component which is the core of the RT-PRLA system and is made up of the batch module and the real-time module, each responsible for performing computation on batch data and real-time (new) data respectively.

The batch module and real-time module correspond to the batch layer and speed layer of the Lambda architecture. There are several big data application technologies used to develop these layers. A common big data technology for the batch layer is Hadoop, and a common technology for the speed layer is the Apache Spark. Newer technologies such as Apache Spark can also be used to implement both batch and speed layer of the Lambda architecture in a unified manner.

This research however uses a different approach to both the batch module and real-time module. It uses probabilistic programming to design probabilistic models used to perform reasoning on a specific application or problem domain. Probabilistic programming is one of the main concepts of this research, and this study uses the approach as explained by Pfeffer (2016).

A probabilistic model is a generic knowledge about a domain, encoded in probabilistic terms (cf. Section 2.3.1). Two probabilistic models are developed as the major constituent of the server component. One for each module in the server component, let’s call them *batch* model and *real-time* model. These models are then used to perform computations on data to produce the views as relates to the Lambda architecture. The batch model is used to perform pre-computation on all dataset in the master database to produce batch views while the real-time model processes the new data to produce real-time views. More details are present in the next section.

The views are subsequently stored in the databases. The batch views are stored in the batch-view database while the real-time views are stored in the real-time-view database.

Subsequently these views and inference algorithms are used to effectively respond to queries. Results to queries are usually in the form of a probabilistic score.

4.4 Design

In this section, a more descriptive explanation of the design of a real-time probabilistic reasoning system is presented. This explanation throws more light on the functionality of the server, the storage and the feeder.

4.4.1 The Feeder

The feeder is essential to the RT-PRLA system as it acts as the doorway that feeds data into the system. It is also responsible for aggregating data from multiple sources, compressing the data and filtering irrelevant and/or duplicate data entries. In a nutshell, the feeder is responsible for data pipelining into the system and backpressure management.

There are several big data streaming applications to choose from when implementing the feeder. Some examples are Apache Kafka, Akka, Apache Flink, and Apache Spark.

4.4.2 The Storage

Storage is very essential to the system. In section 4.3.1.2, it was mentioned that this RT-PRLA system is made up of a storage component which comprises four databases. The four databases are evenly distributed to cater for the two modules of the server component.

An essential quality of a big data system is its ability to answer as many queries as possible. To achieve this, data should be stored in its raw form, must be immutable and permanently true (Marz & Warren, 2015). Data immutability means that once a data entry is stored, it must never be updated or deleted. This enables data simplicity and prevents human errors. Each entry of data is therefore always true because of data immutability.

The master database is used to store the immutable, raw and permanently true data, thus making it the only source of truth for the system. Thus, the master database must support random writes.

The batch module of the server components also reads data from the master database for processing. These reads happen at a scheduled interval and all data in the master database are retrieved for processing. This means that the master databases must also support bulk reads.

After the batch computation on the batch module, the result of the computation known as the batch views is stored in the batch-view database. Thus, the batch-view database must support bulk updates/writes. The batch views stored in the batch-view database are used to respond to queries. Again, the batch-view database must support random reads.

The pseudo-master and real-time-view databases are used by the real-time module to process recent data. The data storage responsibility of the real-time module is very challenging. The real-time module requires low latency random reads from the pseudo-master database and low latency random writes to the real-time-view database. Thus, both databases must support random reads and random writes with low latency to enable quick response to queries and quick update of real-time views.

All four databases must also be scalable. This means that all four databases must efficiently scale with an increase in the data they store. Again, all four databases must be fault-tolerant. They must continue to function as normal if there is a hardware failure.

There are open-sourced NoSQL databases that can be used to implement the storage component. Some examples are MongoDB and Cassandra. A choice between a combination of more than one NoSQL databases or just one NoSQL database can be made.

4.4.3 The Server

An introduction to the server component was presented in Section 4.3.1.3. The server component can be considered as the *engine* of the real-time probabilistic reasoning system. As stated earlier, the main functionality of the server component is to perform computation on both batch and real-time data to produce corresponding batch views and real-time views. Thus, there are two parts to the server component – the batch module which is responsible for batch computation, and the real-time module that performs incremental computation on new data.

Computation on the batch module of the server component is done at intervals. The batch module retrieves and performs computation on the entire dataset in the master database. Depending on the size of the dataset in the master database, processing usually takes long to complete.

The real-time module takes care of the high-latency associated with computation in the batch module. It performs computation on new data that was not included during the last pre-computation done in the batch module. Like the batch module, the fundamental objective of the real-time module is to produce views used to respond to queries. There are two approaches to the real-time module. One is to produce views by processing all the recent data yet to be processed in the batch module, the other is to perform incremental computation on new data as they arrive. This research uses the latter to ensure better resource management and low-latency computation.

Section 4.3.2 indicated a major difference between this solution and the majority of big data solutions available. As the name of the solution, *real-time probabilistic reasoning system using Lambda architecture* implies, one of the distinct differences is the employment of probabilistic programming in the design of probabilistic models in the server component instead of using the conventional Hadoop, Spark, or Storm. Another important difference is the usage of Lambda architecture in the probabilistic reasoning system to improve low latency in data computation.

A probabilistic reasoning system comprises two main pairs – probabilistic model and inference algorithm. Probabilistic model is derived from general relevant knowledge in a domain, encoded in quantitative, probabilistic terms. This probabilistic model is now used in combination with an inference algorithm to answer queries given evidence (cf. Section 2.3.1).

Probabilistic models are created using a combination of representational languages such as Bayesian and Markov, and mathematical constructs. This usually requires a high level of technical expertise and can be a difficult task. Thus, the introduction of probabilistic programming which uses the powerful features of a programming language in the design of a probabilistic model (cf. Section 2.3.2) makes it easier to express probabilistic models and enables automatic inference computation on the models to respond to queries.

Since the inception of research in probabilistic programming, several probabilistic programming systems have been developed (cf. Section 2.3.2). Therefore, the server component can be implemented using any of the probabilistic programming systems available.

4.5 Chapter Summary

This chapter provided a detailed design of a real-time probabilistic reasoning system using Lambda architecture (RT-PRLA) and reiterated the need for a real-time probabilistic reasoning system as well as the objectives of this research.

A general overview of the RT-PRLA system was presented, clearly stating the components and the reasoning process of the system. The RT-PRLA consists of three components namely feeder, storage and server components.

The feeder component is solely responsible for data transformation and aggregation. It also acts a data pipeline into the RT-PRLA system.

The storage component acts as the data store of the system. There are four different databases in the storage component – the master database, the pseudo-master database, the batch-view database and the real-time-view database.

The server component is the engine of the RT-PRLA system. The server component is made up of two modules called the batch module and the real-time module. It is responsible for all batch and incremental computations on data. The batch module reads all the data in the master database and performs computation on it to produce batch views which are stored in the batch-view database. The real-time module incrementally reads data from the pseudo-master database which contains recent data not included in the batch computation and performs incremental computation to produce real-time views. Real-time views are subsequently stored in the real-time-view database.

Computations in the batch and real-time modules are implemented using the concept of probabilistic programming. Thus, batch and real-time views correspond to probabilistic models.

CHAPTER FIVE
KOGNITOR, AN EXPOSITORY CASE STUDY

In the previous chapter, the design concept of a real-time probabilistic reasoning system using Lambda architecture was presented. This design concept is based on the six (6) identified design objectives all directed at achieving the second aim of this research.

Chapter 4 follows the design activity as proposed by Peffers et al. (2007), and corresponds to the third step (cf. Section 3.4.3). The next steps according to the research process are to demonstrate (cf. Section 3.4.4) and evaluate (cf. Section 3.4.5) the proposed artefact.

This chapter is dedicated to demonstrating and evaluating the RT-PRLA system. The first section of this chapter gives a short introduction to the context of the case study. Section 5.2 presents the problem domain of this demonstration and a description of the artefact called *Kognitor*. In Section 5.3, the tools and technologies used in the construction of Kognitor are described following the design concept presented in the previous chapter. This section also demonstrated the construction of a probabilistic model using probabilistic programming in the server component of the RT-PRLA system. The implementation of Lambda architecture in the three components of the RT-PRLA system is also described in Section 5.3.

Section 5.4 presents an evaluation of the design artefact using the experimental evaluation method in a simulated environment. The results of this evaluation are also presented in tabular format in this section. Section 5.5 summarises this chapter.

5.1 Introduction

The usefulness of an artefact is made evident when it is applied to one or more instances of the problem in a relevant domain (Peffers et al., 2007). This demonstration can be actualized as a case study, proof, experimentation or simulation. In any of the methods of demonstration, A. R. Hevner et al. (2004) stated that the requirements must be clearly defined by the business context.

In this case, the business context chosen for the demonstration is predicting the outcome of a football match between two teams in the English Premiership League (EPL). This context was selected because it allows the presentation of the different capabilities of the design concept presented in chapter 4.

The novelty of this research is basically in its design concept, and not necessarily in the implementation or construction of the artefact. Vaishnavi et al. (2004) supported that the implementation of an artefact does not need to involve creativity outside the state-of-practice; however, the originality should be in the design. Thus, the main purpose of this case study is to show the practicality and usefulness of the design concept of a RT-PRLA system.

The central part of the design concept is the server component (cf. Section 4.4.3) that provides the novel capabilities of using probabilistic programming on big data in real-time. A secondary design concept is the implementation of the RT-PRLA using Lambda architecture.

5.2 Problem Domain

The output of a design science research should solve or improve the solution to an applicable business problem. The design concept of the RT-PRLA system aims at simplifying the decision-making process, especially in uncertain circumstances by providing real-time probabilistic responses to queries on big data.

The relevant problem domain in this case is prediction and inference or deduction. The use case in this chapter handles predicting a win between two football teams in the EPL contest. This case study called Kognitor, can also be used to infer the reason(s) of an outcome of a football match.

5.3 Tools and Technologies

This section describes the tools and technologies used in the implementation of Kognitor. One of the objectives of this design is to use already existing open-source big data tools and architectures in the design of an artefact for cost-effectiveness. Thus, all the tools and technologies used in the development of Kognitor are off the shelf systems.

5.3.1 Feeder

The feeder component of Kognitor is implemented using Akka. Akka is a powerful, flexible and resilient tool for building scalable, high-performance, concurrent and distributed real-time event processing and message-driven applications (Lightbend Inc, 2010; Akka, 2011).

Akka is based on the actor model of computation. The actor model was introduced in 1973 by (Hewitt et al., 1973). According to Hewitt (2011) and Hewitt (2012), computation in this model is distributed, as computational devices called *actors* transmit and receive messages asynchronously in no predefined order. One of the advantages of this model is the ease of creating concurrent and distributed systems.

In Akka, actors are created in an actor system. Each actor has an address, a mailbox, a state and a behaviour. An actor can send messages to another actor through its unique address. These messages are stored in an actor’s mailbox for processing. Messages in an actor’s mailbox are processed one at a time (Akka, 2011; Mishra, 2017). Akka is widely used in the development of a variety of distributed applications and frameworks (Rosà et al., 2016).

Akka also provides a solution for stream processing called *Akka Streams API*. The Akka Streams API is based on the Akka actor-model and provides the fundamental principles for developing back-pressured streaming applications (Mishra, 2017; Lightbend Inc., 2019).

In the feeder component of Kognitor, a simulated data repository was used as the data source and an Akka actor (called the `feeder-actor`) was used to implement a pipeline between the repository and Kognitor. The `feeder-actor` retrieves data from the repository at a predetermined interval and sends it to another actor (`persistData-actor`) to store in the storage component of Kognitor. As soon as the `persistData-actor` is done, it sends a *learn* message to the `learn-actor`. The `learn-actor` is responsible for initiating the learning process on the speed layer of Kognitor. The `feeder-actor` is also responsible for initiating the learning process in the batch layer by calling a `batchlearn-actor`.

For this thesis, data for only two teams were considered. Thus, learning on two teams was initiated sequentially using two actors, one per team.

5.3.2 Storage

Kognitor uses Apache Cassandra as its storage component. This means that all four databases were implemented using Cassandra. According to The Apache Software Foundation (2015), Cassandra has the best support for replication across multiple datacentres, and this provides low latency reads and writes. Apache Cassandra is also fault-tolerant and a right choice for scalability.

As mentioned in chapter 4, the storage component of Kognitor is made up of four (4) databases, two of which forms part of the batch layer of this Lambda architecture implementation. The other two also forms part of the speed layer (cf. Section 4.3.1.2). The two databases in the batch layer are called *k_master* (for the master database) and *k_batchview* (for the batch-view database). In the speed layer, the pseudo-master database is called *k_pseudomaster* while the real-time-view database is called *k_realtimeview*.

The k_master database is made up of four tables namely *team*, *rating*, *form* and *fixture*. These tables hold the immutable data as specified in the Lambda architecture. The team table holds generic information about a football team, the rating table stores the rating of a football team, the form table holds the last six (6) performances of a football team, and the fixture table stores data about a football match event. Data in the rating, form and fixture tables are timestamped. The k_pseudomaster database is like the k_master database and contains the same tables (see APPENDIX A: STORAGE).

There are additional ‘feeder’ tables in the k_master database which are not part of the system’s source of truth but were used to implement the simulation of data flow into the system (cf. Section 5.3.2).

After computation on data in the k_master and k_pseudomaster databases, results are stored in the respective databases. The k_batchview database holds results of computation done on data in k_master database while the k_realtimeview database holds results of computation on data in the k_pseudomaster database. Both k_batchview and k_realtimeview databases have a similar table called *teamprobability* (see APPENDIX A: STORAGE).

5.3.3 Server

The server component is the core of Kognitor. It is responsible for performing computations on the data stored in the master and pseudo-master databases using probabilistic programming. As indicated in chapter 4, the server component is divided into two modules – the batch module and the real-time module.

The batch module and real-time module are developed using Figaro. Figaro is a probabilistic programming language that enables easier creation of probabilistic models using the powerful features of the Scala programming language.

Probabilistic models are represented as *Figaro* models consisting of elements. Queries to a probabilistic reasoning system are directed at specific elements of interest. According to Pfeffer (2016), a Figaro model requires four main features:

- The elements/variables in the model.
- The relationships between elements in the model.
- The functional forms of the relationships.
- The numerical parameters of the functional forms.

Each of these components is discussed in detail with respect to Kognitor.

Elements in the Figaro Model.

In choosing elements for our football model in Kognitor, the properties of a football team that indicates whether they win in a competition are considered. These properties are also known as variables. In Figaro, these variables have types/classes such as String, Double, Boolean, etc. Some of these properties include the following:

- Team financial strength.
- Team's position in the English Premiership table.
- Team's rating.
- Team's form.
- Team's home ground advantage.
- Individual performance of players in the team.
- Team's manager.

To keep the application relatively simple, the following elements are selected for Kognitor:

- `hasGoodRating` – An element whose value is true if a team has good rating. Teams rating is between 0.0 and 10.0. The type of this element is Boolean.
- `hasGoodForm` – A element whose value is true if a team has won most of the last six (6) played games. This is a Boolean element.
- `hasHomeGroundAdvantage` – A Boolean element whose value is true if a team has home-ground advantage. This means that the team has either won or at least drew most of the games played on their home-ground.
- `isWinner` – A Boolean element whose value is true if the team won a game.

The dependencies between the elements

After choosing the elements in the model, the next step is to define the dependencies between these elements. Probabilistic reasoning is about using dependencies between elements. Dependency between two elements are established if knowledge of one element influences the other element (Pfeffer, 2016).

There are two types of dependencies – the *directed* dependencies which usually go from one element to the other, and the *undirected* dependencies which models the instance where the direction of influence is unknown. Direct dependencies are encoded using *Bayesian networks* while the undirected dependencies are modelled using *Markov networks*.

For our football model, the dependencies between the chosen elements are based on the following simplifying decisions:

- A team's form, rating and home-ground advantage determines if they win or lose a game.

The dependency model is depicted in Figure 5.1.

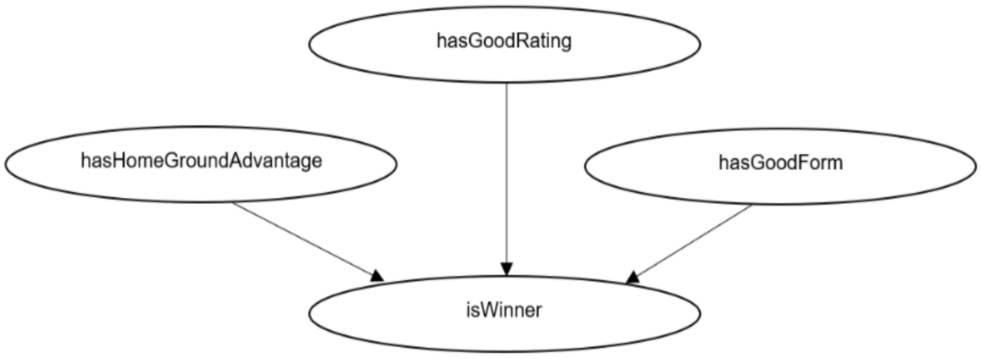


Figure 5.1: Bayesian dependency model for Kognitor elements

The elements are represented as nodes in the dependency diagram, and there is an edge between one node to another if the second node is influenced by the first node. This kind of network is known as a *Naïve Bayes' model*.

The functional forms of the dependencies

In Figaro, functional forms are the element class constructors that are used to build the model. Functional forms express the dependencies between the elements in a model in probabilistic terms without numerical values.

In our model, we have selected the elements and have established the dependencies between the elements. Now, let's determine the functional forms these dependencies take. Let's start with the elements that do not depend on any other variable.

The `hasGoodRating` element has a Boolean type with true or false values. This value depends on the actual rating probability of a team which is derived from the range 0 to 10. In this range, 0 – 5.9 corresponds to bad, 6 – 6.9 corresponds to average, and 7 – 10 corresponds to good. In this case, the `If` construct in Figaro is used to specify the functional form for `hasGoodRating`. The `If` construct is one way to specify a compound element in Figaro. `If` constitutes a test, a then clause (element) and an else element (Pfeffer, 2016). The functional form of the `hasGoodRating` element is as follows:

```
val highRating = Flip(goodRatingProbability)
val lowRating = Flip(badRatingProbability)
val hasGoodRating = If(isWinner, highRating, lowRating)
```

The `hasGoodForm` element is also a Boolean type with true or false values. This element depends on the probability that a team has won at least four (4) out of their last six (6) games. The `If` class is used to represent this functional form in Figaro:

```
val highForm = Flip(goodFormProbability)
val lowForm = Flip(badFormProbability)
val hasGoodForm = If(isWinner, highForm, lowForm)
```

The same goes for the `hasHomeGroundAdvantage` element, it is also represented using the `If` construct:

```
val hasGoodHead2Head = Flip(goodHead2HeadProbability)
val hasBadHead2Head = Flip(badHead2HeadProbability)
val hasHomeGroundAdvantage = If(isWinner,
hasGoodHead2Head, hasBadHead2Head)
```

The last element `isWinner` is of a Boolean type and a deterministic variable. This means that it is fully dependent on the `hasHomeGroundAdvantage`, `hasGoodRating` and `hasGoodForm` elements. The `Flip` construct is used here to specify the functional form. `Flip` takes a Double argument between 0 and 1 inclusive, which represents the probability that the element's value is true (Pfeffer, 2016). The functional form of `isWinner` is defined as follows:

```
val isWinner = Flip(winProbability)
```

This represents the probability that a team wins a match.

Numerical parameters

This is the final component in the design of a probabilistic model. The functional forms have parameters. These parameters are assigned numerical values. Care must be taken to assign valid values for the functional forms.

In our case, initial numerical values are assigned to the parameters before any data is seen. The `Beta` distribution is used to derive the initial probabilities. The `Beta` distribution has two numerical parameters which represents the number of times two outcomes have been observed, plus 1. After data have been passed through the model through the feeder component, these numerical values are 'learned' or calculated from the data.

The elements `highRating` and `lowRating` are given initial values that represent the probability that a team's rating is greater than or equal to 6.0 and less than 0.6 respectively. Subsequently, after data is known, this numerical value for `highRating`, `lowRating` and `hasGoodRating` is learned from the data.

The functional form of `hasHomeGroundAdvantage` is specified by the `If` construct. Again, initial parameters are given to the probabilities using the `Beta` distribution. In this case, we assume that a team has won as least four (4) games out of six (6) games in their home ground as the initial value for `hasGoodHead2Head`. The opposite assumption is made for the initial value of `hasBadHead2Head`. Thereafter, when data is known, this numerical parameter is calculated.

The numerical parameter for `hasGoodForm` element is also derived using a `Beta` distribution. An assumption that a team has won 4 or more games in their last six (6) games is used to derive the initial parameter for `highForm`. The initial value of `lowForm` is derived based on the assumption that a team has lost at least 4 games out of their last six (6) games. Subsequently, when data is known, these parameters are calculated.

Initial numerical parameter for `isWinner` is derived based on the assumption that a team has won as much games as they have lost.

Putting the elements, the dependencies between the elements, the functional forms of the elements and their numerical parameters all together provides a Figaro (football) model for our case study. This football model is used for learning as well as reasoning on data.

The learning process involves using the data in the master and pseudo-master databases as training data to produce football models used in the reasoning process. However, the learning process uses a prior football model which is made up of initial parameters (assumptions) as mentioned earlier. The learning process transforms the prior football model into reasoning football models using a learning algorithm and the training data. Figaro provides a learning algorithm called *expectation maximization* (EM) which is used as the learning algorithm in Kognitor.

A point to remember is that learning happens in both the batch and real-time modules of the server component in Kognitor, and at different intervals. In the batch module, learning happens at a scheduled interval and it uses all the data available in the master database as training data. Learning in the real-time module happens on new data as soon as it enters the system.

The result of learning in the batch component is the football model used to perform batch reasoning while the football model produced from learning in the real-time module is used to perform real-time reasoning. The batch and real-time views are stored in the batch-view database and real-time-view database respectively.

Just like the learning process, reasoning process is done in the batch and real-time modules to provide a probabilistic score to queries. However, instead of a learning algorithm, reasoning uses any inference algorithm. Figaro has a collection of built-in inference algorithms. In Kognitor, the variable elimination (VE) algorithm is used as the inference algorithm.

Queries to Kognitor are targeted towards an element of interest. The default element of interest is the `isWinner` element. The inference algorithm uses the football models (batch view and real-time views) to respond to queries.

5.4 Testing

This section focuses on the testing of the Kognitor system to show the effectiveness of the design concept.

5.4.1 Test System

The evaluation method to demonstrate the usefulness of this research was selected to be experimental (cf. Section 3.4.5). This involves simulation in an appropriate environment for which Kognitor is designed for. Thus, a test environment consistent with the design science research methodology is designed.

The test environment provides a common interface between the simulation and Kognitor (the system to be tested) as is obtainable in the real world. The simulation is developed using Cassandra database running on a Docker container. The simulation environment houses three data objects namely *ratingfeeder*, *fixturefeeder* and *formfeeder* (cf. Section 5.3.2). These feeder objects contain timed data about a team; the ratingfeeder contains ratings of a team at a given time, the fixturefeeder contains data about football matches between two teams and the formfeeder contains data on a team’s form at a given time.

Data is exposed to Kognitor from the database in the simulation through a RESTful webservice implemented using the Play framework. Kognitor consumes the data in the testbed through a data pipeline implemented by the Akka `feeder-actor` (cf. Section 5.3.1a.5.3.1). The system stores and performs learning computation on the data. As mentioned in Section 5.3.1a.5.3.3, learning happens in two places: the batch layer and the speed layer. The feeder-actor is scheduled to read data from the simulation in a 1-day interval for this test. Learning Process in the speed layer happens immediately data is read into Kognitor. The feeder-actor is also scheduled to initial batch learning computation every 7 days. Results of computation are also stored in the system ready for query.

The various components of Kognitor are developed using off-the-shelf technologies. The server component of Kognitor is implemented using the Scala programming language, Figaro, Akka and the Play framework while the storage component is implemented using Cassandra database on a Docker container. Communication between the various components of Kognitor is enabled via RESTful web services.

Kognitor also exposes RESTful web services that accepts queries and responds with probabilistic answers.

5.4.2 Test Results

As stated earlier, two teams from the EPL namely Manchester United and Chelsea were selected for this test. The test was done on the teams' previous EPL games for last season (2017/18) and current season (2018/19); that is Manchester vs Chelsea and vice versa. Both teams had played three (3) games, two (2) of which were in the last season.

5.4.1.1 Learning

On the first run, the teams' data for the first game were injected into Kognitor. Learning computation happened immediately on the speed layer. This learning was repeated at least five (5) times to get an approximate learning duration. It was discovered that learning on new data takes approximately 1 second (see Table 5.1). After the repeated test on speed layer, the batch learning process was initiated. Again, it was run five (5) times to get an estimate of its duration. The batch learning process and the real-time learning process took approximately the same amount of time as learning was on the same data size.

For the second game, real-time learning in the speed layer happened at approximately the same amount of time in the first run. This is because learning computation in the real-time module of Kognitor is incremental; it only learns on the new data and increments/updates the real-time view. In this test, this incremental update is implemented by taking an average of the new real-time view and the old real-time view. However, for the batch layer, the amount of data size has increased and learning computation in the batch module uses all the data in the master database. Thus, the learning process took longer compared to the first run; approximately 3 seconds (see Table 5.2).

Table 5.1: First Run Learning Time in Seconds

Manchester United	Chelsea	Total Time
0.48	0.994	1.474
0.416	0.537	0.993
0.383	0.534	0.917
0.503	0.878	1.381
0.581	0.757	1.338
Average Time		**1.2126**

Table 5.2: Second Run Learning Time (on batch module) in Seconds

Manchester United	Chelsea	Total Time
1.353	1.204	2.557
1.605	1.997	3.602
1.784	1.615	3.399
1.689	4.985	3.674
1.889	1.483	3.372
Average Time		**3.3208**

Table 5.3: Third Run Learning Time (on batch module) in Seconds

Manchester United	Chelsea	Total Time
2.948	3.222	6.17
3.152	3.22	6.372
2.625	2.942	5.567
2.577	2.687	5.264
3.047	2.587	5.634
Average Time		**5.8014**

Learning on all three games in the batch module also took longer (See Table 5.3). A point to remember is that as soon as the result from the batch reasoning is complete, the corresponding real-time view is discarded and replaced with the batch view. This is following the guidelines of Lambda architecture. See APPENDIX B: TEST LEARNING RESULTS for learning results.

5.4.2.1 Reasoning

Kognitor is designed such that there must be available views (batch and real-time) for a successful reasoning. Kognitor exposes a RESTful web service endpoint that accepts a reasoning request on a team. A reasoning request is directed at the `isWinner` element. There are three (3) reasoning options available in Kognitor which is enabled by the implementation of Lambda architecture. Thus, a reasoning request can be directed at the batch module, the real-time module or both batch and real-time modules.

Table 5.4: Reasoning Time in Seconds

	Real-time Module	Batch Module	Real-time & Batch Modules
Time 1	0.038	0.032	0.053
Time 2	0.043	0.031	0.06
Time 3	0.04	0.041	0.061
Time 4	0.029	0.030	0.063
Time 5	0.054	0.022	0.058
Average	**0.0408**	**0.312**	**0.059**

Response time for the individual modules is depicted in Table 5.4. Reasoning with the three (3) options were carried out on one team repeatedly for five (5) times. The reasoning takes less time than the learning processes. This achieves low latency in big data processing.

5.5 Chapter Summary

This chapter focuses on the demonstration and evaluation of the artefact design described in Chapter 4. This is in fulfilment of the fourth (cf. Section 3.4.4) and fifth (cf. Section 3.4.5) activities of the DSRM process as outlined by Peffers et al. (2007).

The result of a design science research should demonstrate its usefulness in the problem domain using an appropriate demonstration method and must be evaluated using a defined evaluation method to see how well it solves the problem in accordance to the specified objectives.

This chapter started with an introduction that lead to the choice of a problem domain – prediction and inference. Then a case study called Kognitor was implemented to demonstrate the design concept described in Chapter 4. The various big data tools and technologies used in this demonstration were described in the context of the different components of the RT-PRLA system. The server component which forms the core of Kognitor was described with emphasis on the creation of probabilistic models using Figaro, a probabilistic programming system, and the effective combination of big data tools to implement the Lambda architecture for real-time response to queries.

Kognitor was evaluated using the experimental evaluation method and tested with simulated data. This evaluation was against the design objectives listed in Section 4.2 of Chapter 4 and the second research question (R2) presented in Chapter 1 (cf. Section 1.4).

Kognitor used Akka actors to manage data ingestion and back pressure. Computations in Kognitor are divided into two: learning computation and reasoning computation. The learning computation happens periodically in the batch module and real-time module respectively, and this is also initiated using Akka actors. The learning computation uses a prior probabilistic model on data and a learning algorithm to produce views used for the reasoning computation. This satisfies the design objective DR4 (cf. Section 4.2).

Reasoning computation is like the learning computation with two differences: (1) Reasoning computation uses an inference algorithm on the result of the learning computation to respond to queries. Responses to queries are probabilistic scores. (2) There is a choice of reasoning option for the user; the user can reason on the batch layer, the real-time layer or both.

Response time for both learning and reasoning were present in the testing section of this chapter. The response time for the reasoning computation is of more importance to this research. Thus, low latency was achieved using probabilistic programming and an artistic combination of big data tools to implement Lambda architecture.

CHAPTER SIX
RESEARCH EVALUATION

Chapter 1 presented the aim and objectives of this study which were reiterated in Chapter 4 as the design requirement for the RT-PRLA system. This chapter presents an assessment of the research process used in this study to achieve the design for the RT-PRLA in the context of the aim and objectives as stated in chapter 1.

6.1 Introduction

The output of a design science research is a design artefact which should be relevant, and the process of construction and evaluation of the artefact should involve rigorous methods (Hevner et al., 2004). This chapter measures the research activities and results against the guidelines by A. R. Hevner et al. (2004) to justify this research as a design science research.

6.2 Hevner's DSR Guidelines

This section outlines the seven (7) guidelines proposed by A. R. Hevner et al. (2004), and how this research implemented or adhered to each guideline.

6.2.1 Design as an Artefact

The first guideline states that the result of a design science research should be an artefact developed to address a problem. This artefact can be in the form of either construct, model, methods, instantiations or a combination of those. The implementation and application of the artefact should be effectively described.

This research resulted in an artefact in the form of an instantiation – a real-time reasoning system using Lambda architecture (RT-PRLA). Chapter 4 presented the design concept of the RT-PRLA system. The RT-PRLA system was designed to advance the implementation of big data solutions using probabilistic programming.

Chapter 4 explained the three (3) components of the RT-PRLA system namely the feeder component, the storage component and the server component. The server component is the engine of the RT-PRLA system and includes the novel approach pertinent to this research.

6.2.2 Problem Relevance

The second guideline emphasizes the need for relevance. The design artefact should provide a solution to a relevant problem. "The objective of design-science research is to develop technology-based solutions to important and relevant business problems." (Hevner et al., 2004: 83).

The problem space or specific environment determines the relevance of a DSR endeavour (cf. Chapter 3). In this research, the problem space is specific to decision-makers. Decision making is part of, and central to the existence of every business.

Section 1.2 and Section 1.3 of Chapter 1 provided the background and motivation to this research and stated the problem statement that addresses the relevance of this research. Chapter 2 also emphasized the importance of this research after a systematic literature review. The design concept of the RT-PRLA system is to aid decision-makers especially in uncertain situations by providing real-time probabilistic reasoning on big data.

6.2.3 Design Evaluation

Guideline 3 talks about the evaluation of the design artefact. This is an important step in the design science research process. "The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods." (Hevner et al., 2004: 83). Evaluation determines the usefulness of an artefact in the business environment.

Chapter 5 presented the implementation of a case study called Kognitor. Kognitor was evaluated using simulation method. Simulation is one of the acceptable ways to evaluate a designed artefact as suggested by A. R. Hevner et al. (2004). In addition to the simulation process in chapter 5, there was continuous evaluation during the design process to achieve the design detail decision as supported by Vaishnavi et al. (2004).

6.2.4 Contribution

The fourth guideline addresses the need for a design science research to have a comprehensible and justifiable contribution to the body of knowledge. The contribution can be in the form of a design artefact, design foundations, and/or design methodologies.

The primary contribution of this research is the design artefact, the RT-PRLA system. The RT-PRLA system is the result of the artistic combination of the probabilistic programming concept and big data technologies using Lambda architecture to improve decision-making tasks especially during uncertainty.

A secondary contribution of this research can be seen in the contribution towards the body of knowledge in the field of big data, real-time big data processing and Lambda architecture. The developed case study in chapter 5 demonstrated how the RT-PRLA system can be implemented using existing off-the-shelf big data tools and probabilistic programming languages on commodity hardware, thus making it cost-effective.

6.2.5 Research Rigor

Guideline 5 describes the importance of rigor in design science research. A. R. Hevner et al. (2004) pointed out that rigorous methods should be used in the construction and evaluation of the design artefact. This rigor is achieved through the effective use of theoretical foundations in the domain and research methodologies.

Chapter 2 presented the knowledge base relevant to this research. In chapter 3, the research paradigm, methodology, methods and techniques were chosen with reasons based on the research pyramid of Jonker & Pennink (2009). Chapter 3 also adopted the design science research process of Peffers et al. (2007). Each step in the design science research process was based on knowledge derived from the literature review in chapter 2. Also, the various methods and techniques used to design, implement and evaluate the design artefact were gathered from literature. Some examples are the Lambda architecture and probabilistic programming technique.

6.2.6 Design as a Search Process

A. R. Hevner et al. (2004) describes the design science research process as an iterative (search) process that seeks to solve relevant business problems using the appropriate knowledge base.

The RT-PRLA system is the result of a search process. In chapter 2, current big data tools and techniques were identified and a systematic literature review was performed. Chapter 4 presented a list of design objectives established because of a search process.

6.2.7 Communication of Research

The final guideline is the effective communication of the research to the relevant audience. A. R. Hevner et al. (2004, p. 83), states that “Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.”

This thesis is a major communication documentation of this research. Again, this research or part of it was presented at various occasions during the research process. During the early stages of this research, a proposal was presented to a mixed audience comprising senior academics, postgraduate students and industry officials at the Department of Information Technology and Faculty of Informatics and Design, CPUT.

The section titled “Publications from this Research” outlines a list of all conferences/journals where papers from this research were presented.

6.3 Chapter Summary

The different steps and stages in this research were revisited and measured against Hevner's et al. (2004) design science research guidelines. This is to ensure that this research adequately complies with the design science research process.

All the search and design process that resulted in the development of the RT-PRLA system were documented in this thesis. The RT-PRLA system was demonstrated and evaluated using the experimental method.

CHAPTER SEVEN
DISCUSSION & CONCLUSION

The main goal of this research was introduced in chapter 1 which is to develop a real-time big data technology that uses probabilistic reasoning and Lambda architecture to output response in real-time.

The first section of this chapter summarises this study and presents an introduction to this chapter. Section 7.2 presents the findings of this research with respect to the research aims. Section 7.3 contains a summary of this research and Section 7.4 indicates the direction for further research.

7.1 Introduction

This research tackles the problem associated with real-time probabilistic processing of both real-time and historic big data. Probabilistic reasoning involves the development of a probabilistic model which is passed through an inference algorithm to produce probabilistic results. However, due to the identified hurdles associated with the development of probabilistic models using well established probability theories like Bayesian Network or Markov models, the programming and machine learning community proposed a novel concept called probabilistic programming.

To determine the extent of influence probabilistic programming has in the processing of big data, this study included a systematic literature review of big data system implemented using probabilistic programming. The garnered information indicated the possible effective influence of probabilistic programming in big data processing. Thus, this research's aim and objectives were re-established to show the relevance of this study. An artefact was developed and evaluated using design science research methodology. The design process to achieve the artefact was also evaluated using documented procedures to demonstrate research rigor.

This chapter wraps up this study and provides findings and answers to the research questions.

7.2 Research Findings

The general purpose of this research is to find out existing big data solutions that implemented probabilistic programming in processing data, and to improve decision making by developing a real-time probabilistic reasoning system. This aim resulted in two research questions for the research (cf. Section 1.4).

7.2.1 Answers to Research Questions

This section provides answers to the research questions outlined in chapter 1. These answers are based on the fulfilment of the research objectives stated in chapter 1.

RQ 1 What are the existing big data solutions implemented using probabilistic programming?

To answer this question, a systematic literature review was conducted which is in line with the first objective of this research. However, before the systematic review, chapter 2 presented a literature review on the various knowledge bases relevant to this research.

The result of the systematic literature review indicated one big data solution called InferSpark that uses probabilistic programming (cf. Section 2.4). InferSpark is a probabilistic programming framework built on Apache Spark. The developers of InferSpark claimed that at the time of publication, InferSpark was the only solution that implemented big data processing using probabilistic programming (Zhao et al., 2017).

RQ 2 How to achieve low latency in big data processing?

During the systematic literature review process, the only solution that was found was also evaluated. The review identified a limitation in InferSpark: InferSpark could not scale out on a distributed framework. Again, InferSpark is built on Apache Spark which supports several machine learning algorithms, but they do not support user defined models. Other findings identified by the designer of InferSpark is the claim that InferSpark is the only big data solution that uses probabilistic programming.

These key findings were utilized in the response to the second research question. They also emphasized the need and usefulness of a real-time probabilistic reasoning system (cf. Section 2.4.4.4).

This research proposes the use of existing big data solutions, the concept of probabilistic programming and Lambda architecture to develop a tool to handle real-time big data processing. Data computation is solely handled using probabilistic programming implemented using Figaro, and the real-time response is achieved using Akka actors and Lambda architecture developed by Marz & Warren (2015).

The design concept was demonstrated with a case study: Kognitor. The crux of the design concept is that the server component is implemented using Figaro that allows for easy creation of user-defined probabilistic models used along with inference algorithms provided by Figaro to respond to queries in real-time.

The evaluation of Kognitor demonstrated low latency in the processing of data (cf. Section 5.4).

7.3 Summary

This thesis reports a research project regarding the design of a real-time probabilistic reasoning system using Lambda architecture. The inspiration behind this research is the need for an automated real-time system that enhances decision-making especially during uncertainty. This need was identified after a systematic literature review was conducted.

Decision-makers are required to make crucial decisions, most times in the face of uncertainty. This decision-making process entails the use of experience and recent events to arrive at a decision. This means that reasoning over historic data and real-time data to produce real-time response is imperative.

To improve decision-making in times of uncertainty, this research proposed the real-time probabilistic reasoning (RT-PRLA) system using Lambda architecture. The design concept of the RT-PRLA system is centred around probabilistic programming for data processing and Lambda architecture to achieve real-time computation on both historic and real-time data.

The core of the RT-PRLA system is the server component which is made up of two modules: the batch module and the real-time module. A probabilistic model is implemented on the server component. Both modules of the server components are responsible for retrieving data from their corresponding data stores (master database and pseudo-master database) and creating corresponding post parameters (batch views and real-time views) which are like the serving layer and speed layer of the Lambda architecture. These views are used with the probabilistic model and inference algorithm to respond to queries in real-time.

Another contribution of the RT-PRLA system is in the reuse of existing open-source big data technologies and commodity hardware to develop a cost-effective big data solution.

7.4 Future Work

The scope of this research was tied to the research goals which were to find the number of existing big data technologies that use probabilistic programming to handle big data problems, and to design a real-time big data solution using probabilistic programming and Lambda architecture.

Thus, the focus was mainly on the probabilistic reasoning on big data using probabilistic programming and Lambda architecture. Other human-related components such as the user interface that deals with users' interaction with the system were designated as out of scope for this research. Therefore, a possible direction for future work could be in the design of the user interface to the RT-PRLA system.

Two out of the highlighted design objectives: DR5 and DR6 (cf. Section 4.2) were not implemented in this study. DR5 proposes that the system should implement a fully automated compressing and filtering mechanism when ingesting data. In this study, an already clean simulated data was used. DR6 also suggests that the system should be generic: used in different problem domains. These are areas for further research.

Furthermore, the experimental evaluation method was used to assess the RT-PRLA system. This evaluation was demonstrated using a test system and simulation with real data. This is a limitation factor. Simulation was selected as a suitable method for evaluating the system; however, other evaluation methods could present further findings to the design concept of the RT-PRLA system. Consequently, more testing using different evaluation methods is required.

REFERENCES

Ábrahám, E. & Havelund, K. 2016. Some recent advances in automated analysis. *International Journal on Software Tools for Technology Transfer*, 18(2): 121–128. http://link.springer.com/10.1007/s10009-015-0403-0.

Acharya, S. & Biswal, M.P. 2011. Solving probabilistic programming problems involving multi-choice parameters. *OPSEARCH*, 48(3): 217–235. http://link.springer.com/10.1007/s12597-011-0053-2.

Adar, E. & Re, C. 2007. Managing uncertainty in social networks. *IEEE Data Eng. Bull*, 30(2): 15–22. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.397&amp;rep=rep1&amp;type=pdf.

Adhianto, L., Banerjee, S., Fagan, M., Krentel, M., Marin, G., Mellor-Crummey, J. & Tallent, N.R. 2010. HPCTOOLKIT: Tools for performance analysis of optimized parallel programs. *Concurrency Computation Practice and Experience*, 22(6): 685–701.

Akka. 2011. Akka: Build powerful reactive, concurrent, and distributed applications more easily. https://akka.io/ 10 November 2017.

Alon, N. 2013. Paul Erdős and probabilistic reasoning. In *Bolyai Society Mathematical Studies*. 11–33.

Apache Software Foundation. 2015. Apache Storm. http://storm.apache.org/ 13 February 2018.

Apache Software Foundation. 2010. S4 Incubation Status - Apache Incubator. http://incubator.apache.org/projects/s4.html 13 February 2018.

Apache Software Foundation. 2016. Welcome to Apache^{TM} Hadoop®! http://hadoop.apache.org/ 14 May 2016.

ApacheFlume. 2016. Welcome to Apache Flume — Apache Flume. https://flume.apache.org/ 4 November 2017.

ApacheKafka. 2017. Apache Kafka. https://kafka.apache.org/ 13 February 2018.

ApacheSamza. 2016. Samza. http://samza.apache.org/ 9 November 2017.

Artikis, A., Etzion, O., Feldman, Z. & Fournier, F. 2012. Event processing under uncertainty. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems - DEBS '12*. New York, New York, USA: ACM Press: 32–43. http://dl.acm.org/citation.cfm?doid=2335484.2335488.

Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A.S. & Buyya, R. 2015. Big Data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79–80: 3–15. http://www.sciencedirect.com/science/article/pii/S0743731514001452.

Astakhov, V. & Chayel, M. 2015. Lambda Architecture for Batch and Real- Time Processing on AWS with Spark Streaming and Spark SQL. , (May): 1–12. https://d0.awsstatic.com/whitepapers/lambda-architecure-on-for-batch-aws.pdf.

Baer, T. 2013. *The Promise of Fast Data – An Inside Look at Oracle Technology*. Ovum.

Bajaber, F., Elshawi, R., Batarfi, O., Altalhi, A., Barnawi, A. & Sakr, S. 2016. Big Data 2.0 Processing Systems: Taxonomy and Open Challenges. *Journal of Grid Computing*, 14(3): 379–405. http://dx.doi.org/10.1007/s10723-016-9371-1.

Barlow, M. 2013. *Real-Time Big Data Analytics - Emerging Architecture*. 1st ed. Carlifornia: O'Reilly.

Barnes, T.J. 2013. Big data, little history. *Dialogues in Human Geography*, 3(3): 297–302. http://dhg.sagepub.com/lookup/doi/10.1177/2043820613514323.

Baskerville, R., Pries-Heje, J. & Venable, J. 2009. Soft design science methodology. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology - DESRIST '09*. New York, New York, USA: ACM Press: 11. http://portal.acm.org/citation.cfm?doid=1555619.1555631.

Bayesia. 2018. BayesiaLab Introduction. http://www.bayesia.com/introduction 27

February 2018.

Benbasat, I. & Zmud, R. 1999. Empirical research in information systems: the practice of relevance. *MIS quarterly*, 23(1): 3–16. http://www.jstor.org/stable/249403.

Bendler, J., Wagner, S., Brandt, T. & Neumann, D. 2014. Taming Uncertainty in Big Data. *Business & Information Systems Engineering*, 6(5): 279–288. http://link.springer.com/10.1007/s12599-014-0342-4.

Bhadani, A.K. & Jothimani, D. 2016. Big Data: Challenges, Opportunities and Realities. In M. K. Singh & D. . Kumar, eds. *Effective Big Data Management and Opportunities for Implementation*. Pennsylvania, USA: IGI Global: 1–24. https://arxiv.org/ftp/arxiv/papers/1705/1705.04928.pdf.

Bifet, A. 2013. Mining big data in real time. *Informatica (Slovenia)*, 37(1): 15–20.

Bilandzic, M. & Venable, J. 2011. Towards Participatory Action Design Research: Adapting Action Research and Design Science Research Methods for Urban Informatics. *The journal of Informatics*, 7(3): 1–16. http://ci-journal.net/index.php/ciej/article/view/786/804.

Borkar, V.R., Carey, M.J. & Li, C. 2012. Big data platforms. *XRDS: Crossroads, The ACM Magazine for Students*, 19(1): 44. http://dl.acm.org/citation.cfm?doid=2331042.2331057.

Boyd, D. & Crawford, K. 2012. Critical Questions for Big Data. *Information, Communication & Society*, 15(5): 37–41.

Boyd, D. & Crawford, K. 2011. Six Provocations for Big Data. *SSRN Electronic Journal*: 1–17. http://www.ssrn.com/abstract=1926431.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M. & Khalil, M. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4): 571–583. http://dx.doi.org/10.1016/j.jss.2006.07.009.

Brown, B., Chui, M. & Manyika, J. 2011. *Are you ready for the era of Big Data?* McKinsey & Company.

Budgen, D. & Brereton, P. 2006. Performing systematic literature reviews in software engineering. *Int. Conf. Soft. Engin.*: 1051. http://portal.acm.org/citation.cfm?doid=1134285.1134500.

Cambridge Dictionary. 2016. Big meaning in the Cambridge English dictionary. *Dictionary.cambridge.org*. http://dictionary.cambridge.org/dictionary/english/big 13 November 2016.

Cambridge English Dictionary. 2011. research Meaning in the Cambridge English Dictionary. *Cambridge English Dictionary*. https://dictionary.cambridge.org/dictionary/english/research?q=Research 11 March 2018.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M.A., Li, P. & Riddell, A. 2016. Stan : A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1): 1–32.

Chen, J., Chen, Y., Du, X., Li, C., Lu, J., Zhao, S. & Zhou, X. 2013. Big data challenge: a data management perspective. *Frontiers of Computer Science*, 7(2): 157–164. http://www.scopus.com/inward/record.url?eid=2-s2.0-84876005101&partnerID=40&md5=8bc0d2c00d407e15a6d37d1df34e5f42.

Chen, M., Mao, S. & Liu, Y. 2014. Big Data: A Survey. *Mobile Networks and Applications*, 19(2): 171–209. http://link.springer.com/10.1007/s11036-013-0489-0.

Chen, S., Li, W., Li, M., Zhang, X. & Min, Y. 2014. Latest Progress and Infrastructure Innovations of Big Data Technology. In *2014 International Conference on Cloud Computing and Big Data*. IEEE: 8–15. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7062865.

Choi, J., Nazareth, D.L. & Jain, H.K. 2010. Implementing Service-Oriented Architecture in Organizations. *Journal of Management Information Systems*, 26(4): 253–286.

Collins Dictionary. 2018. Practice definition and meaning | Collins English Dictionary. https://www.collinsdictionary.com/dictionary/english/reasoning 22 February 2018.

Conrady, S. & Jouffe, L. 2013. *Knowledge Discovery in the Stock Market with Bayesian Network*. bayesia.us.

Creswell, J. 2009. *Research design: Qualitative, quantitative, and mixed method approaches*. 2nd ed. Sage Publications.

Creswell, J.W. 2007. *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*. Second. Sage Publications.

Creswell, J.W. 2013. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 4th ed. Sage Publications.

Crotty, M. 1998. *The foundations of social research: Meaning and perspective in the research process*. London: Sage Publications.

Davenport, T.H., Barth, P. & Bean, R. 2012. How ‘ Big Data ’ Is Different. *MIT Sloan Management Review*, 54(1): 43–46.

Dean, B.Y.J. & Ghemawat, S. 2010. MapReduce: a flexible data processing tool. *Communications of the ACM*, 53(1): 72–77. http://portal.acm.org/citation.cfm?doid=1629175.1629198%5Cnhttp://dl.acm.org/citation.cfm?id=1629198.

Dean, J. & Ghemawat, S. 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1): 107. http://dl.acm.org/citation.cfm?id=1327452.1327492%5Cnhttp://portal.acm.org/citation.cfm?doid=1327452.1327492.

Delen, D. & Demirkan, H. 2013. Data, information and analytics as services. *Decision Support Systems*, 55(1): 359–363. http://dx.doi.org/10.1016/j.dss.2012.05.044.

Demchenko, Y., Grosso, P., de Laat, C. & Membrey, P. 2013. Addressing big data issues in Scientific Data Infrastructure. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE: 48–55. http://ieeexplore.ieee.org/document/6567203/.

DeWitt, D. & Gray, J. 1992. Parallel database systems: the future of high performance database systems. *Communications of the ACM*, 35(6): 85–98. http://portal.acm.org/citation.cfm?doid=129888.129894.

Dijcks, J. 2012. Oracle: Big data for the enterprise. *Oracle White Paper*, (June): 16. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Oracle+:+Big+Data+for+the+Enterprise#0.

Dobre, C. & Xhafa, F. 2014. Parallel Programming Paradigms and Frameworks in Big Data Era. *International Journal of Parallel Programming*, 42(5): 710–738. http://link.springer.com/10.1007/s10766-013-0272-7.

Domingos, P. & Richardson, M. 2007. Markov Logic: A Unifying Framework for Statistical Relational Learning. *Introduction to Statistical Relational Learning*: 339–367. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.8085&rep=rep1&type=pdf%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.8085%5Cnhttp://books.google.com/books?hl=en&lr=&id=lSkIewOw2WoC&oi=fnd&pg=PA339&dq=Markov+Logic:+A+Unifying+Fr.

Dries, A., Kimmig, A., Meert, W., Renkens, J., Van den Broeck, G., Vlasselaer, J. & De Raedt, L. 2015. ProbLog2: Probabilistic Logic Programming. In *Joint European Conference on Machine Learning and Knowledge in Databases ECML PKDD 2015*. Springer, Cham: 312–315. http://link.springer.com/10.1007/978-3-319-23461-8_37.

Durbin, R., Eddy, S., Krogh, A. & Mitchison, G. 1998. *Biological sequence analysis*. New York: Cambridge University Press. http://eisc.univalle.edu.co/cursos/web/material/750068/1/6368030-Durbin-Et-Al-Biological-Sequence-Analysis-CUP-2002-No-OCR.pdf.

Evans, J.R. & Lindner, C.H. 2012. Business Analytics: The Next Frontier for Decision Sciences. http://www.cbpp.uaa.alaska.edu/afef/business_analytics.htm 10 February 2018.

Fan, J., Han, F. & Liu, H. 2014. Challenges of Big Data analysis. *National Science Review*, 1(2): 293–314. http://nsr.oxfordjournals.org/cgi/doi/10.1093/nsr/nwt032.

Fan, W. & Bifet, A. 2013. Mining big data. *ACM SIGKDD Explorations Newsletter*, 14(2): 1. http://dl.acm.org/citation.cfm?doid=2481244.2481246.

Fernandez, R.C., Pietzuch, P., Kreps, J., Narkhede, N., Rao, J., Koshy, J., Lin, D., Riccomini, C. & Wang, G. 2015. Liquid: Unifying Nearline and Offline Big Data Integration. *Conference on Innovative Data Systems Research*.

Figueira, J., Greco, S. & Ehrogott, M. 2005. *Multiple Criteria Decision Analysis: State of the Art Surveys*. New York, NY: Springer New York. http://link.springer.com/10.1007/b100605.

Fitz-enz, J. 2009. Predicting people: From metrics to analytics. *Employment Relations Today*, 36(3): 1–11. http://doi.wiley.com/10.1002/ert.20255.

Ganchev, I., Ji, Z. & O'Droma, M. 2016. A conceptual framework for building a mobile services' recommendation engine. In *2016 IEEE 8th International Conference on Intelligent Systems (IS)*. IEEE: 285–289. http://ieeexplore.ieee.org/document/7737435/.

Gandomi, A. & Haider, M. 2015. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2): 137–144. http://dx.doi.org/10.1016/j.ijinfomgt.2014.10.007.

Gantz, B.J. & Reinsel, D. 2011. Extracting Value from Chaos State of the Universe : An Executive Summary. *IDC iView*, (June): 1–12. http://idcdocserv.com/1142.

Gebara, F.H., Hofstee, H.P. & Nowka, K.J. 2015. Second-Generation Big Data Systems. *Computer*, 48(1): 36–41. http://ieeexplore.ieee.org/document/7030152/.

Gedik, B., Andrade, H., Wu, K.-L., Yu, P.S. & Doo, M. 2008. SPADE: The System S Declarative Stream Processing Engine. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*. New York, New York, USA: ACM Press: 1123. https://access.korea.ac.kr/link.n2s?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-55649105542&lang=ko&site=eds-live&scope=site.

Geerts, G.L. 2011. A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems*, 12(2): 142–151. http://dx.doi.org/10.1016/j.accinf.2011.02.004.

Gehr, T., Misailovic, S. & Vechev, M. 2016. PSI: Exact Symbolic Inference for Probabilistic Programs. In S. Chaudhuri & A. Farzan, eds. *International Conference on Computer Aided Verification*. Springer, Cham: 62–83. http://psisolver.org/.

Ghahramani, Z. 2015. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553): 452–459. http://www.nature.com/articles/nature14541.

Ghemawat, S., Gobioff, H. & Leung, S.-T. 2003. The Google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP '03*. New York, New York, USA: ACM Press: 29. http://portal.acm.org/citation.cfm?doid=945445.945450.

Glass, R.L. 1999. The Loyal Opposition - On Design. *IEEE Software*, 16(2): 104–103. http://ieeexplore.ieee.org/document/754066/.

Gonzalez, J. 2012. *Parallel and Distributed Systems for Probabilistic Reasoning*. Carnegie Mellon University.

Goodman, N.D., Mansinghka, V., Roy, D., Bonawitz, K. & Tenenbaum, J.B. 2008. Church: a language for generative models. In *Proceedings of 24th Conference on Uncertainty in Artificial Intelligence*. 220–229. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=9536E59A560463C98699658C9DAA50DC?doi=10.1.1.151.7160&rep=rep1&type=pdf%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.151.7160%5Cnhttp://arxiv.org/abs/1206.3255.

Goodman, N.D. & Stuhlmüller, A. 2014. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org 25 March 2018.

Gordon, A.D., Graepel, T., Rolland, N., Russo, C., Borgstrom, J. & Guiver, J. 2014. Tabular: a schema-driven probabilistic programming language. *Proceedings of* *the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL '14*, (1): 321–334. http://dl.acm.org/citation.cfm?doid=2535838.2535850.

Gordon, A.D., Henzinger, T.A., Nori, A. V. & Rajamani, S.K. 2014. Probabilistic programming. In *Proceedings of the on Future of Software Engineering - FOSE 2014*. New York, New York, USA: ACM Press: 167–181. http://dl.acm.org/citation.cfm?doid=2593882.2593900.

Gregor, S. & Hevner, A.R. 2013. Positioning and Presenting Design Science Types of Knowledge in Design Science Research. *MIS Quarterly*, 37(2): 337–355.

Guba, E.G. 1990. *The paradigm dialog*. Sage Publications. http://www.jstor.org/stable/3340973.

Haenni, R. 2005. Towards a unifying theory of logical and probabilistic reasoning. *Isipta*, 5(4): 1.

Han Hu, Yonggang Wen, Tat-Seng Chua & Xuelong Li. 2014. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *IEEE Access*, 2: 652–687. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6842585.

Hansen, D. 2013. *Introduction to Oracle Fast Data*. California: Oracle.

Harrison, G. 2014. Real-Time Big Data With the Lambda Architecture. *Database Trends & Applications*, 28(5): 31–31.

Hasani, Z., Kon-Popovska, M. & Velinov, G. 2014. Lambda Architecture for Real Time Big Data Analytic. *ICT Innovations*: 133–143.

Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A. & Ullah Khan, S. 2015. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47: 98–115. http://dx.doi.org/10.1016/j.is.2014.07.006.

Healy, M. & Perry, C. 2000. Comprehensive criteria to judge validity and reliability of qualitative research within the realism paradigm. *Qualitative Market Research: An International Journal*, 3(3): 118–126. http://www.emeraldinsight.com/doi/10.1108/13522750010333861.

Hevner, A. & Chatterjee, S. 2010. Design Science Research in Information Systems. In *Design Research in information Systems*. Integrated Series in Information Systems. Boston, MA: Springer US: 9–22. http://desrist.org/design-research-in-information-systems.

Hevner, A.R., March, S.T., Park, J. & Ram, S. 2004. Design Science in Information Systems Research. *MIS Quarterly*, 28(1): 75–105. http://dblp.uni-trier.de/rec/bibtex/journals/misq/HevnerMPR04.

Hewitt, C. 2011. Actor Model of Computation. *Inconsistency Robustness 2011*: 1–25. http://arxiv.org/abs/1008.1459.

Hewitt, C. 2012. What is Computation? Actor Model versus Turing's Model. In *A Computable Universe*. WORLD SCIENTIFIC: 159–185. http://www.worldscientific.com/doi/abs/10.1142/9789814374309_0009.

Hewitt, C., Bishop, P. & Steiger, R. 1973. A universal modular ACTOR formalism for artificial intelligence. In *IJCAI'73 Proceedings of the 3rd international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ©1973: 235–245.

Hicks, M. 2014. What is probabilistic programming? (The Programming Languages Enthusiast). http://www.pl-enthusiast.net/2014/09/08/probabilistic-programming/ 25 March 2018.

Hildreth, C.R. & Aytac, S. 2007. Recent library practitioner research: A methodological analysis and critique. *Journal of Education for Library and Information Science*, 48(3): 236–258.

Hirzel, M., Schneider, S. & Gedik, B. 2017. SPL: An Extensible Language for Distributed Stream Processing. *ACM Transactions on Programming Languages and Systems*, 39(1): 1–39. http://dx.doi.org/10.1145/3039207.

Huang, H.H. & Liu, H. 2014. Big data machine learning and graph analytics: Current state and future challenges. In *2014 IEEE International Conference on Big Data (Big Data)*. IEEE: 16–17. http://ieeexplore.ieee.org/document/7004471/.

Iivari, J. & Venable, J.R. 2009. Action Research and Design Science Research. In

Proceedings of 17th European Conference on Information Systems. Verona, Italy: 2711–2723.

Jacobs, A. 2009. The Pathologies of Big Data. *Communications of the ACM*, 52(8): 36–44.

Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J.M., Ramakrishnan, R. & Shahabi, C. 2014. Big data and its technical challenges. *Communications of the ACM*, 57(7): 86–94. http://libezproxy.open.ac.uk/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=96868411&site=eds-live&scope=site.

Jambi, S. & Anderson, K.M. 2017. Engineering Scalable Distributed Services for Real-Time Big Data Analytics. In *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE: 131–140. http://ieeexplore.ieee.org/document/7944930/.

Järvelin, K. & Vakkary, P. 1990. Content analysis of research articles in library and information science. *Library and Information Science Research*, 12(4): 395–421.

Jin, X., Wah, B.W., Cheng, X. & Wang, Y. 2015. Significance and Challenges of Big Data Research. *Big Data Research*, 2(2): 59–64. http://dx.doi.org/10.1016/j.bdr.2015.01.006.

Jonker, J. & Pennink, B. 2009. *The Essence of Research Methodology*. Berlin, Heidelberg: Springer Berlin Heidelberg. http://link.springer.com/10.1007/978-3-540-71659-4.

Katal, A., Wazid, M. & Goudar, R.H. 2013. Big data: Issues, challenges, tools and Good practices. In *2013 Sixth International Conference on Contemporary Computing (IC3)*. IEEE: 404–409. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6612229.

Khan, N., Yaqoob, I., Hashem, I.A.T., Inayat, Z., Mahmoud Ali, W.K., Alam, M., Shiraz, M. & Gani, A. 2014. Big Data: Survey, Technologies, Opportunities, and Challenges. *The Scientific World Journal*, 2014: 1–18. http://www.hindawi.com/journals/tswj/2014/712826/.

Kim, G.-H., Trimi, S. & Chung, J.-H. 2014. Big-data applications in the government sector. *Communications of the ACM*, 57(3): 78–85. http://search.proquest.com/docview/1516150205?accountid=34461.

Kiran, M., Murphy, P., Monga, I., Dugan, J. & Baveja, S.S. 2015. Lambda architecture for cost-effective batch and speed big data processing. In *2015 IEEE International Conference on Big Data (Big Data)*. IEEE: 2785–2792. http://ieeexplore.ieee.org/document/7364082/.

Kitchenham, B. & Charters, S. 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3. *Engineering*, 45(4ve): 1051.

Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O.P., Turner, M., Niazi, M. & Linkman, S. 2010. Systematic literature reviews in software engineering-A tertiary study. *Information and Software Technology*, 52(8): 792–805. http://dx.doi.org/10.1016/j.infsof.2010.03.006.

Kitchin, R. & Lauriault, T.P. 2015. Small data in the era of big data. *GeoJournal*, 80(4): 463–475. http://dx.doi.org/10.1007/s10708-014-9601-7.

Köhler, M., Kaniovskyi, Y. & Benkner, S. 2015. Towards adaptive execution strategies for large-scale and real-time data analytics. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*: 447–454.

Kraska, T., Talwalkar, A., Duchi, J., Griffith, R., Franklin, M. & Jordan, M. 2013. MLbase : A Distributed Machine-learning System. *6th Biennial Conference on Innovative Data Systems Research (CIDR'13)*.

Kreps, J. 2014. Questioning the Lambda Architecture. *O'Reilly*: 1–10. https://www.oreilly.com/ideas/questioning-the-lambda-architecture 18 October 2017.

Kuhn, T.S. 1970. The Structure of Scientific Revolutions. *International Encyclopedia of Unified Science*, II(2): 210. http://www.jstor.org/stable/10.2307/2183664.

Kutty, A.D., Kumar Shee, H. & Pathak, R.. 2007. Decision-making. *Monash Business*

Review, 3(3): 8–9. http://publications.epress.monash.edu/doi/abs/10.2104/mbr07056.

Labrinidis, A. & Jagadish, H. V. 2012. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12): 2032–2033. http://dl.acm.org/citation.cfm?doid=2367502.2367572.

Lake, B.M., Ullman, T.D., Tenenbaum, J.B. & Gershman, S.J. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40: 72. https://www.cambridge.org/core/product/identifier/S0140525X16001837/type/journal_article.

Lämmel, R. 2008. Google's MapReduce programming model — Revisited. *Science of Computer Programming*, 70(1): 1–30. http://linkinghub.elsevier.com/retrieve/pii/S0167642307001281.

Landset, S., Khoshgoftaar, T.M., Richter, A.N. & Hasanin, T. 2015. A survey of open source tools for machine learning with big data in the Hadoop ecosystem. *Journal of Big Data*, 2(1): 24. http://www.journalofbigdata.com/content/2/1/24.

Laney, D. 2001. *3D Data Managment: Controlling Data Volume, Velocity and Variety*. Meta Group.

Lee, K.-H., Lee, Y.-J., Choi, H., Chung, Y.D. & Moon, B. 2012. Parallel data processing with MapReduce. *ACM SIGMOD Record*, 40(4): 11–20. http://portal.acm.org/citation.cfm?id=2094118.

Li, J., Tao, F., Cheng, Y. & Zhao, L. 2015. Big Data in product lifecycle management. *International Journal of Advanced Manufacturing Technology*, 81(1–4): 667–684.

Lightbend Inc. 2019. Introduction - Akka Documentation. https://doc.akka.io/docs/akka/current/stream/stream-introduction.html 8 June 2019.

Lightbend Inc. 2010. Akka, Actor-based message-driven runtime | @lightbend. https://www.lightbend.com/akka 10 November 2017.

Lin, J., Leu, F. & Chen, Y. 2015. ReHRS: A Hybrid Redundant System for Improving MapReduce Reliability and Availability. In 187–209. http://link.springer.com/10.1007/978-3-319-09177-8.

Lin, J., Milligan, I., Wiebe, J. & Zhou, A. 2017. Warcbase: Scalable analytics infrastructure for exploring web archives. *Journal on Computing and Cultural Heritage*, 10(4): 1–30. http://dl.acm.org/citation.cfm?doid=3129537.3097570.

Liu, G., Zhu, W., Saunders, C., Gao, F. & Yu, Y. 2015. Real-time Complex Event Processing and Analytics for Smart Grid. *Procedia Computer Science*, 61: 113–119. http://dx.doi.org/10.1016/j.procs.2015.09.169.

Liu, S., Duffy, A.H.B., Whitfield, R.I. & Boyle, I.M. 2010. Integration of decision support systems to improve decision support performance. *Knowledge and Information Systems*, 22(3): 261–286. http://link.springer.com/10.1007/s10115-009-0192-4.

Liu, X., Iftikhar, N. & Xie, X. 2014. Survey of real-time processing systems for big data. In *Proceedings of the 18th International Database Engineering & Applications Symposium on - IDEAS '14*. New York, New York, USA: ACM Press: 356–361. http://dl.acm.org/citation.cfm?doid=2628194.2628251.

Lohr, S. 2012. The Age of Big Data. *The New York Times*: 1–5.

Lorentz, A. 2013. Big Data, Fast Data, Smart Data. http://www.wired.com/insights/2013/04/big-data-fast-data-smart-data/ 12 May 2016.

Luger, G. & Chakrabarti, C. 2008. Knowledge-Based Probabilistic Reasoning from Expert Systems to Graphical Models. *Handbook of Probability: Theory and Applications*: 2–22. http://www.cs.unm.edu/~luger/23-Luger-Chakrabarti.pdf.

Lunn, D., Spiegelhalter, D., Thomas, A. & Best, N. 2009. The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, 28(25): 3049–3067. http://doi.wiley.com/10.1002/sim.3680.

Luo, L. & McKinney, M. 2015. JAL in the Past Decade: A Comprehensive Analysis of Academic Library Research. *The Journal of Academic Librarianship*, 41(2): 123–129. http://dx.doi.org/10.1016/j.acalib.2015.01.003.

Mackenzie, N. & Knipe, S. 2006. IIER 16: Mackenzie and Knipe - research dilemmas: Paradigms, methods and methodology. *Issues In Educational Research*. http://www.iier.org.au/iier16/mackenzie.html 3 June 2016.

Madden, S. 2012. From Databases to Big Data. *IEEE Internet Computing*, 16(3): 4–6. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6188576.

Manning, C.D. & Raghavan, P. 2009. An Introduction to Information Retrieval. In *Online*. 1. http://dspace.cusat.ac.in/dspace/handle/123456789/2538.

Mansinghka, V., Selsam, D. & Perov, Y. 2014. Venture: a higher-order probabilistic programming platform with programmable inference. : 1–78. http://arxiv.org/abs/1404.0099.

March, S.T. & Smith, G.F. 1995. Design and natural science research on information technology. *Decision Support Systems*, 15(4): 251–266. http://linkinghub.elsevier.com/retrieve/pii/0167923694000412.

Margara, A., Urbani, J., van Harmelen, F. & Bal, H. 2014. Streaming the Web: Reasoning over dynamic data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 25: 24–44. http://dx.doi.org/10.1016/j.websem.2014.02.001.

Markus, L.M., Majchrzak, A. & Gasser, L. 2002. A design theory for systems that support emergent knowledge processes. *MIS Quarterly*, 26(3): 179–212. http://www.jstor.org/stable/pdf/4132330.pdf?refreqid=excelsior%3Aebbf0c52e1641656d48980f6912a8a89.

Marz, N. & Warren, J. 2015. *Big Data: Principles and best practices of scalable real-time data systems*. New York: Manning. http://nathanmarz.com/about/.

De Mauro, A., Greco, M. & Grimaldi, M. 2015. What is big data? A consensual definition and a review of key research topics. In *AIP Conference Proceedings*. 97–104. http://aip.scitation.org/doi/abs/10.1063/1.4907823.

McAfee, A. & Brynjolfsson, E. 2012. Big data: the management revolution. *Harvard business review*, 90(10): 59–68.

Merriam-Webster. 2016. Definition of Big by Merriam-Webster. *Merriam-Webster.com*. http://www.merriam-webster.com/dictionary/big 13 November 2016.

Merriam-Webster. 2018. Definition of Reasoning by Merriam-Webster. https://www.merriam-webster.com/dictionary/reasoning 22 February 2018.

Microsoft. 2013. The Big Bang: How the Big Data Explosion Is Changing the World. *Microsoft News Center*: 1–35. https://news.microsoft.com/2013/02/11/the-big-bang-how-the-big-data-explosion-is-changing-the-world/ 28 March 2018.

MIKE 2.0. 2018. Information Asset Concept - MIKE2.0, the open source methodology for Information Development. *MIKE2.0*. http://mike2.openmethodology.org/wiki/Big_Data_Definition 27 January 2018.

Milch, B., Marthi, B., Russel, S., Sontag, D., Ong, D.L. & Kolobov, A. 2007. Probabilistic models with unknown objects. *Statistical Relational Learning*: 352.

Miloslavskaya, N. & Tolstoy, A. 2016. Big Data, Fast Data and Data Lake Concepts. *Procedia Computer Science*, 88: 300–305. http://dx.doi.org/10.1016/j.procs.2016.07.439.

Mishne, G., Dalton, J., Li, Z., Sharma, A. & Lin, J. 2013. Fast data in the era of big data: Twitter’s real-time related query suggestion architecture. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*: 1147–1158.

Mishra, P. 2017. Streaming and the Actor Model – Akka Streams! | Packt Hub. *Packt*. https://hub.packtpub.com/streaming-and-actor-model-akka-streams/ 8 June 2019.

Mohanty, H., Bhuyan, P. & Chenthati, D. 2015. *Big Data*. H. Mohanty, P. Bhuyan, & D. Chenthati, eds. New Delhi: Springer India. http://link.springer.com/10.1007/978-81-322-2494-5.

Mohapatra, S.K., Sahoo, P.K. & Wu, S.-L. 2016. Big data analytic architecture for intruder detection in heterogeneous wireless sensor networks. *Journal of Network and Computer Applications*, 66: 236–249.

http://dx.doi.org/10.1016/j.jnca.2016.03.004.

Myers, M.D. & Venable, J.R. 2014. A set of ethical principles for design science research in information systems. *Information & Management*, 51(6): 801–809. http://dx.doi.org/10.1016/j.im.2014.01.002.

Nair, L.R., Shetty, S.D. & Deepak Shetty, S. 2017. Streaming Big Data Analysis for Real-Time Sentiment based Targeted Advertising. *International Journal of Electrical and Computer Engineering (IJECE)*, 7(1): 402. http://www.iaescore.com/journals/index.php/IJECE/article/view/6293.

Narayanan, P., Carette, J., Romano, W., Shan, C. & Zinkov, R. 2016. Probabilistic Inference by Program Transformation in Hakaru (System Description). In 62–79. http://link.springer.com/10.1007/978-3-319-29604-3_5.

NIST. 2018. CSRC Topics - big data | CSRC. https://csrc.nist.gov/Topics/Technologies/big-data 3 February 2018.

Norsys. 2013. Norsys - Netica Application. http://www.norsys.com/netica.html 27 February 2018.

Nunamaker, J.F., Chen, M. & Purdin, T.D.M. 1990. Systems Development in Information Systems Research. *Journal of Management Information Systems*, 7(3): 89–106. http://www.tandfonline.com/doi/full/10.1080/07421222.1990.11517898.

Okoli, C. & Schabram, K. 2010. A Guide to Conducting a Systematic Literature Review of Information Systems Research. *SSRN Electronic Journal*, 10(26): 1–51. http://sprouts.aisnet.org/10-26.

Oxford English Dictionaries. 2018. research | Definition of research in English by Oxford Dictionaries. *Oxford English Dictionaries*. https://en.oxforddictionaries.com/definition/research 11 March 2018.

Patel, S. 2015. The Research Paradigm - Methodology, Epistomology and Ontology - Explained in Simple Language. http://salmapatel.co.uk/academia/the-research-paradigm-methodology-epistemology-and-ontology-explained-in-simple-language 11 March 2018.

Peffers, K., Tuunanen, T., Rothenberger, M.A. & Chatterjee, S. 2007. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3): 45–77. http://mesharpe.metapress.com/index/276818W6PN4T5483.pdf%5Cnhttp://mesharpe.metapress.com/openurl.asp?genre=article&id=doi:10.2753/MIS0742-1222240302.

Perera, S. & Suhothayan, S. 2015. Solution patterns for realtime streaming analytics. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems - DEBS '15*. New York, New York, USA: ACM Press: 247–255. http://dl.acm.org/citation.cfm?doid=2675743.2774214.

Pfeffer, A. 2009. *Figaro: An object-oriented probabilistic programming language*. http://www.cs.tufts.edu/~nr/cs257/archive/avi-pfeffer/figaro.pdf%5Cnpapers2://publication/uuid/0E83E526-451F-41EA-ACBE-7150FF7584D4.

Pfeffer, A. 2016. *Practical probabilistic programming*. New York: Manning.

Pfeffer, A. 2007. The Design and Implementation of IBAL: A General-Purpose Probabilistic Language. *Introduction to statistical relational learning*, (1993): 34.

Philip Chen, C.L. & Zhang, C.-Y. 2014. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, 275: 314–347. http://dx.doi.org/10.1016/j.ins.2014.01.015.

Prékopa, A. 2003. Probabilistic Programming. In *Handbooks in Operations Research and Management Science*. 267–351. http://www.cs.cornell.edu/courses/cs4110/2016fa/lectures/lecture33.html 25 March 2018.

Qiu, J., Wu, Q., Ding, G., Xu, Y. & Feng, S. 2016. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1): 67. http://dx.doi.org/10.1186/s13634-016-0355-x.

Rabiner, L.R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2): 257–286. http://ieeexplore.ieee.org/ielx5/5/698/00018626.pdf?tp=&arnumber=18626&isnumber=698%5Cnhttp://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=18626&tag=1%0Ahttp://ieeexplore.ieee.org/document/18626/.

De Raedt, L. & Kersting, K. 2003. Probabilistic logic learning. *ACM SIGKDD Explorations Newsletter*, 5(1): 31. http://portal.acm.org/citation.cfm?doid=959242.959247.

Raghupathi, W. & Raghupathi, V. 2014. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2(1): 3. http://link.springer.com/10.1186/2047-2501-2-3.

Ramírez-Gallego, S., Fernández, A., García, S., Chen, M. & Herrera, F. 2018. Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce. *Information Fusion*, 42: 51–61. http://linkinghub.elsevier.com/retrieve/pii/S1566253517305912.

Rodríguez, P., Kuvaja, P. & Oivo, M. 2014. Lessons learned on applying design science for bridging the collaboration gap between industry and academia in empirical software engineering. *Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry - CESI 2014*: 9–14. http://dl.acm.org/citation.cfm?doid=2593690.2593694.

Rosà, A., Chen, L.Y. & Binder, W. 2016. Profiling actor utilization and communication in Akka. : 24–32.

Roy, D. 2018. Probabilistic Programming. http://www.probabilistic-programming.org/wiki/Home 25 March 2018.

Russom, P. 2013. *Operational Intelligence: Real-Time Business Analytics from Big Data*.

Ryan, B., Scapens, R.W. & Theobold, M. 2002. *Research Method and Methodology in Finance and Accounting*.

Saaty, T.L. 2008. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, 1(1): 83. http://www.inderscience.com/link.php?id=17590.

Sagiroglu, S. & Sinanc, D. 2013. Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE: 42–47. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6567202.

Sampson, A. 2015. Probabilistic Programming. http://adriansampson.net/doc/ppl.html 25 March 2018.

Sato, T. 2008. A glimpse of symbolic-statistical modeling by PRISM. *Journal of Intelligent Information Systems*, 31(2): 161–176. http://link.springer.com/10.1007/s10844-008-0062-7.

Schwandt, T.A. 2001. *Dictionary of qualitative inquiry*. 2nd ed. Thousand Oaks: Sage Publications.

Sein, M.K., Henfridsson, O., Purao, S., Rossi, M. & Lindgren, R. 2011. Action Design Research. *MIS Quarterly*, 35(1): 37. http://www.jstor.org/stable/10.2307/23043488.

Shahrivari, S. 2014. Beyond Batch Processing: Towards Real-Time and Streaming Big Data. *Computers*, 3(4): 117–129. http://www.mdpi.com/2073-431X/3/4/117/.

Shmueli, G. & Koppius, O. 2010. Predictive Analytics in Information Systems Research. *SSRN Electronic Journal*, 35(3): 553. http://www.ssrn.com/abstract=1606674.

Siegel, E. 2013. Predictive analytics. *Analytics-Magazine.Org*: 38–42. http://www.predictiveanalyticsworld.com/book/Predictive Analytics - The Power of Big Data - Siegel in Analytics Magazine - July-August 2013.pdf.

Simon, H.A. 1997. *The sciences of the artificial*. 3rd ed. Cambridge: MIT Press.

Snijder, C., Matzat, U. & Reips, U.-D. 2012. “Big Data”: Big Gaps of Knowledge in the Field of Internet Science. *International Journal of Internet Science*, 7(1): 1–5. http://www.ijis.net/ijis7_1/ijis7_1_editorial.pdf.

Song, S.K., Kim, D.J., Hwang, M., Kim, J., Jeong, D.H., Lee, S., Jung, H. & Sung, W. 2013. Prescriptive analytics system for improving research power. *Proceedings -*

16th IEEE International Conference on Computational Science and Engineering, CSE 2013: 1144–1145.
SQLstream. 2017. SQLstream - A SQL-based Real-time Stream Analytics Platform -. http://sqlstream.com/ 15 February 2018.
Staples, M. & Niazi, M. 2007. Experiences using systematic review guidelines. *Journal of Systems and Software*, 80(9): 1425–1437.
Strode, D.E. & Chard, S.M. 2014. A Proposal for using Design Science in Small-Scale Postgraduate Research Projects in Information Technology. In *2014 International Conference of Teaching, Assessment and Learning (TALE)*. Wellington: IEEE: 242–245.
Strohbach, M., Ziekow, H., Gazis, V. & Akiva, N. 2015. Towards a Big Data Analytics Framework for IoT and Smart City Applications. In 257–282. http://link.springer.com/10.1007/978-3-319-09177-8.
Szolovits, P. & Pauker, S.G. 1978. Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence*, 11(1–2): 115–144. http://linkinghub.elsevier.com/retrieve/pii/0004370278900140.
Tashakkori, A. & Teddlie, C. 2010. *Handbook of mixed methods in social & behavioral research*. Sage Publications.
Taxidou, I. & Fischer, P. 2013. Realtime analysis of information diffusion in social media. *Proceedings of the VLDB Endowment*, 6(12): 1416–1421. http://dl.acm.org/citation.cfm?doid=2536274.2536328.
The Apache Software Foundation. 2015. Apache Cassandra Database. *Cassandra*. http://cassandra.apache.org/ 20 December 2018.
Tolpin, D., van de Meent, J.-W., Yang, H. & Wood, F. 2016. Design and Implementation of Probabilistic Programming Language Anglican. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages - IFL 2016*. New York, New York, USA: ACM Press: 1–12. http://arxiv.org/abs/1608.05263.
Tseng, J.C.C., Gu, J., Wang, P.F., Chen, C., Li, C. & Tseng, V.S. 2016. A scalable complex event analytical system with incremental episode mining over data streams. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE: 648–655. http://ieeexplore.ieee.org/document/7743854/.
Tversky, A. & Kahneman, D. 1975. Judgment under Uncertainty: Heuristics and Biases. In *Utility, Probability, and Human Decision Making*. Dordrecht: Springer Netherlands: 141–162. http://www.springerlink.com/index/10.1007/978-94-010-1834-0_8.
Twardowski, B. & Ryzko, D. 2014. Multi-agent Architecture for Real-Time Big Data Processing. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. IEEE: 333–337. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6928203.
Tyagi, A.K., Priya, R. & Rajeswari, A. 2015. Mining Big Data to Predicting Future. *International Journal of Engineering Research and Applications*, 5(32): 14–21.
Ularu, E.G., Puican, F.C., Apostu, A. & Velicanu, M. 2012. Perspectives on Big Data and Big Data Analytics. *Database Systems Journal*, III(4): 3–14. http://dbjournal.ro/archive/10/10.pdf.
Vaishnavi, V., Kuechler, W. & Petter, S. 2004. Design Science Research in Information Systems. http://www.desrist.org/design-research-in-information-systems/.
Vakali, A., Korosoglou, P. & Daoglou, P. 2016. A multi-layer software architecture framework for adaptive real-time analytics. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE: 2425–2430. http://ieeexplore.ieee.org/document/7840878/.
VANHOVE, T., VAN SEGHBROECK, G., WAUTERS, T., VOLCKAERT, B. & DE TURCK, F. 2016. Managing the Synchronization in the Lambda Architecture for Optimized Big Data Analysis. *IEICE Transactions on Communications*, E99.B(2): 297–306. https://www.jstage.jst.go.jp/article/transcom/E99.B/2/E99.B_2015ITI0001/_article

.
Venable, J.R. 2006. A framework for Design Science research activities. In M. Khosrow-Pour, ed. *Proceedings of the 2006 Information Resources Management Association International Conference*. Washington, DC: Idea Group Publishing: 184–187.
Villari, M., Celesti, A., Fazio, M. & Puliafito, A. 2014. AllJoyn Lambda: An architecture for the management of smart environments in IoT. In *2014 International Conference on Smart Computing Workshops*. IEEE: 9–14. http://ieeexplore.ieee.org/document/7046676/.
Vögler, M., Schleicher, J.M., Inzinger, C. & Dustdar, S. 2017. Ahab: A cloud-based distributed big data analytics framework for the Internet of Things. *Software: Practice and Experience*, 47(3): 443–454. http://doi.wiley.com/10.1002/spe.2424.
Walliman, N. 2001. *Your research project: a step-by-step guide for the first-time researcher*. Sage Publications.
Walls, J.G., Widmeyer, G.R. & El Sawy, O.A. 1992. Building an Information System Design Theory for Vigilant EIS. *Information Systems Research*, 3(1): 36–59.
Wampler, D. 2013. Programming Trends to Watch: Logic and Probabilistic Programming. https://www.thinkbiganalytics.com/2013/03/28/programming-trends-to-watch-logic-and-probabilistic-programming/ 29 March 2018.
Wang, H., Xu, Z., Fujita, H. & Liu, S. 2016. Towards felicitous decision making: An overview on challenges and trends of Big Data. *Information Sciences*, 367–368: 747–765. http://dx.doi.org/10.1016/j.ins.2016.07.007.
Ward, J.S. & Barker, A. 2013. Undefined By Data: A Survey of Big Data Definitions. *arXiv.org*: 2. http://arxiv.org/abs/1309.5821%5Cnpapers3://publication/uuid/63831F5F-B214-46D5-8A86-671042BE993F.
Weber, S. 2010. Design Science Research : Paradigm or Approach? *Proceedings of the 16th Americas Conference on Information Systems*: 1–8.
White, M.D. & Marsh, E.E. 2006. Content Analysis: A Flexible Methodology. *Library Trends*, 55(1): 22–45. http://muse.jhu.edu/content/crossref/journals/library_trends/v055/55.1white.html.
Williams, C. 2007. Research Methods. *Journal of Business & Economic Research*, 5(3): 65–72.
Williams, D. 2018. Predictive coding and thought. *Synthese*, (October 2017). http://link.springer.com/10.1007/s11229-018-1768-x.
Wood, F., van de Meent, J.W. & Mansinghka, V. 2014. A New Approach to Probabilistic Programming Inference. In *17th International Conference on Artificial Intelligence and Statistics*. Reykjavik, Iceland. http://arxiv.org/abs/1507.00996.
Wu, Y., Zheng, L., Heilig, B. & Gao, G.R. 2015. Design and Evaluation of a Novel Dataflow Based Bigdata Solution. *Proceedings of the Sixth International Workshop on Programming Models and Applications for Multicores and Manycores*: 40–48. http://doi.acm.org/10.1145/2712386.2712397.
Xindong Wu, Xingquan Zhu, Gong-Qing Wu & Wei Ding. 2014. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1): 97–107. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6547630.
Yang, F., Merlino, G., Ray, N., Léauté, X., Gupta, H. & Tschetter, E. 2017. The RADStack: Open Source Lambda Architecture for Interactive Analytics. In *Proceedings of the 50th Hawaii International Conference on System Sciences*. 1703–1712. http://hdl.handle.net/10125/41359.
Yang, T. & Shadlen, M.N. 2007. Probabilistic reasoning by neurons. *Nature*, 447(7148): 1075–1080. http://www.nature.com/doifinder/10.1038/nature05852.
Zadeh, L.A. 2003. Toward a perception-based theory of probabilistic reasoning with imprecise probabilities. In *Intelligent Systems for Information Processing*. Elsevier: 3–34. http://linkinghub.elsevier.com/retrieve/pii/B9780444513793500017.
Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. & Stoica, I. 2010. Spark :

Cluster Computing with Working Sets. *HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*: 10.

Zaharia, M., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J. & Venkataraman, S. 2016. Apache Spark: a unified engine for big data processing. *Communications of the ACM*, 59(11): 56–65.

Zakir, J., Seymour, T. & Berg, K. 2015. Big Data Analytics. *Issues in Information Systems*, 16(2): 81–90.

Zhang, Q., Dong, C., Cui, Y. & Yang, Z. 2014. Dynamic uncertain causality graph for knowledge representation and probabilistic reasoning: Statistics base, matrix, and application. *IEEE Transactions on Neural Networks and Learning Systems*, 25(4): 645–663.

Zhao, Z., Pei, J., Lo, E., Zhu, K.Q. & Liu, C. 2017. InferSpark: Statistical Inference at Scale. http://arxiv.org/abs/1707.02047.

Zheng, X., Fu, M. & Chugh, M. 2017. Big data storage and management in SaaS applications. *Journal of Communications and Information Networks*, 2(3): 18–29. http://link.springer.com/10.1007/s41650-017-0031-9.

Zhou, Q., Simmhan, Y. & Prasanna, V. 2013. Towards hybrid online on-demand querying of realtime data with stateful complex event processing. In *2013 IEEE International Conference on Big Data*. IEEE: 199–205. http://ieeexplore.ieee.org/document/6691575/.

APPENDICES

APPENDIX A: STORAGE

Database Schemas

```
CREATE KEYSPACE k_master WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '1'}  AND
durable_writes = true;

CREATE KEYSPACE k_realtimeview WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '1'}  AND
durable_writes = true;

CREATE KEYSPACE k_pseudomaster WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '1'}  AND
durable_writes = true;

CREATE KEYSPACE k_batchview WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '1'}  AND
durable_writes = true;
```

Tables for k_master and k_pseudomaster Schemas

```
CREATE TABLE <schema>.rating (
    teamid text,
    datecreated date,
    rating double,
    PRIMARY KEY (teamid, datecreated)
) WITH CLUSTERING ORDER BY (datecreated ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrateg
y', '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 <schema>.form (
    teamid text,
    datecreated date,
    numberofdraws int,
    numberofloses int,
    numberofwins int,
    PRIMARY KEY (teamid, datecreated)
) WITH CLUSTERING ORDER BY (datecreated ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrateg
y', '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 <schema>.fixture (
    hometeamid text,
    datecreated date,
    awayteamgoals int,
    awayteamid text,
    hometeamgoals int,
    PRIMARY KEY (hometeamid, datecreated)
) WITH CLUSTERING ORDER BY (datecreated ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrateg
y', '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 <schema>.team (
    teamid text PRIMARY KEY,
    teamname 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.SizeTieredCompactionStrateg
y', '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';
```

Table for k_batchview and k_realtimeview Schemas

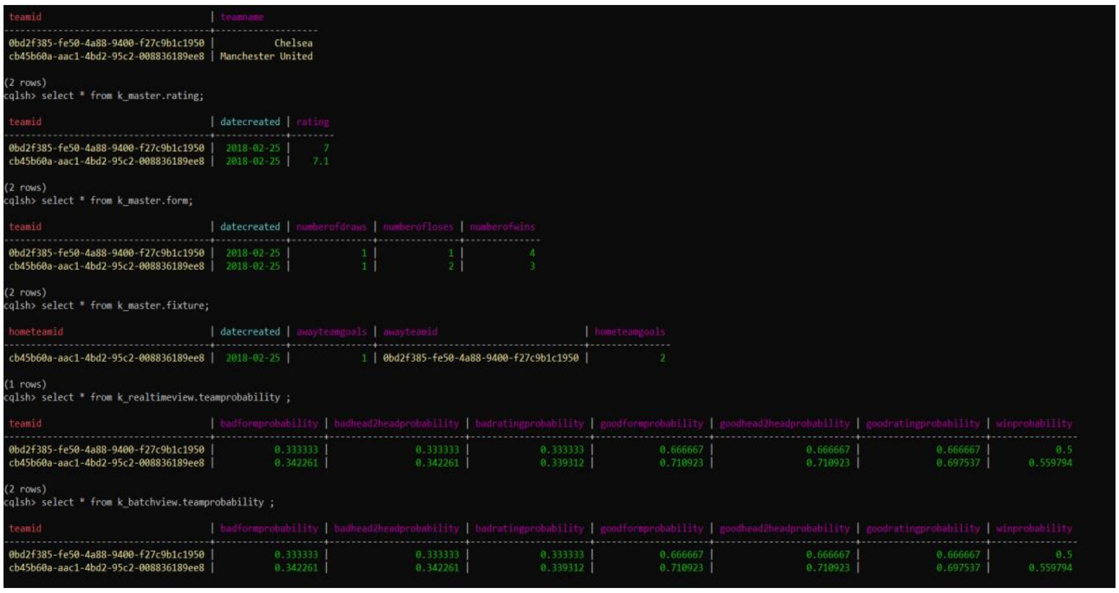
```
CREATE TABLE <schema>.teamprobability (
    teamid text PRIMARY KEY,
    badformprobability double,
    badhead2headprobability double,
    badratingprobability double,
    goodformprobability double,
    goodhead2headprobability double,
    goodratingprobability double,
    winprobability double
) 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.SizeTieredCompactionStrateg
y', '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';
```

Simulation schemas

```
CREATE TABLE k_master.ratingfeeder (
    teamname text,
    datecreated date,
    rating double,
    PRIMARY KEY (teamname, datecreated)
) WITH CLUSTERING ORDER BY (datecreated ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrateg
y', '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 k_master.formfeeder (
    teamname text,
    datecreated date,
    numberofdraws int,
    numberofloses int,
    numberofwins int,
    PRIMARY KEY (teamname, datecreated)
) WITH CLUSTERING ORDER BY (datecreated ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrateg
y', '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 k_master.fixturefeeder (
    teamname text,
    datecreated date,
    awayteamgoals int,
    awayteamname text,
    hometeamgoals int,
    PRIMARY KEY (teamname, datecreated)
) WITH CLUSTERING ORDER BY (datecreated ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrateg
y', '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';
```

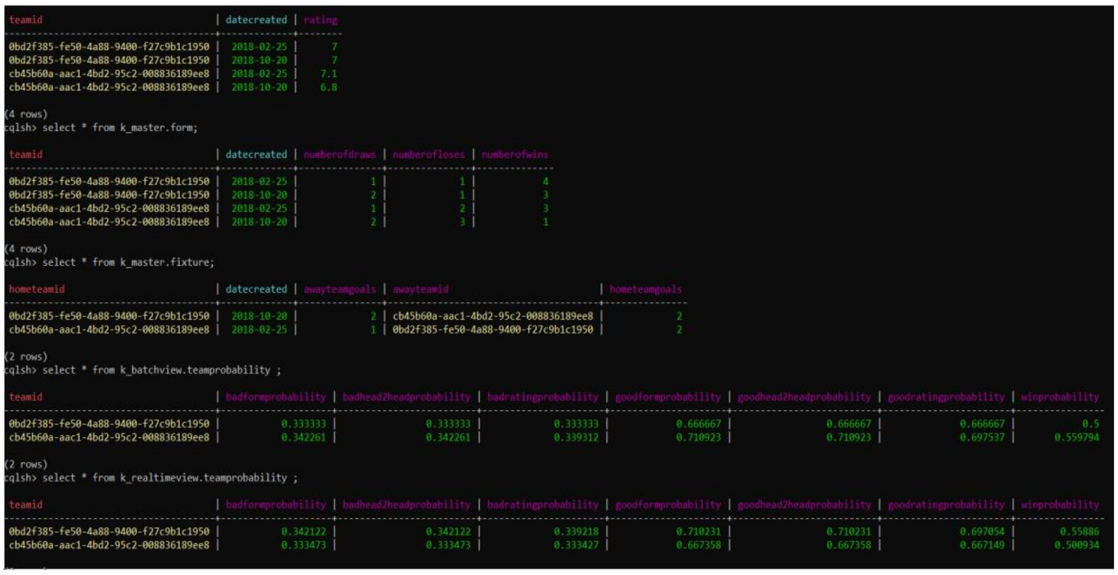
APPENDIX B: TEST LEARNING RESULTS

First Run for First Match (both speed and batch layers).



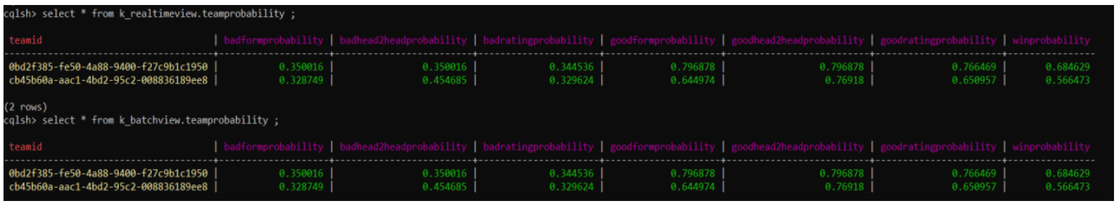
```
 teamid                               | teamname
--------------------------------------+-------------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           Chelsea
 cb45b60a-aac1-4bd2-95c2-008836189ee8 | Manchester United

(2 rows)
cqlsh> select * from k_master.rating;

 teamid                               | datecreated | rating
--------------------------------------+-------------+--------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-02-25 |      7
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |    7.1

(2 rows)
cqlsh> select * from k_master.form;

 teamid                               | datecreated | numberofdraws | numberofloses | numberofwins
--------------------------------------+-------------+---------------+---------------+--------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-02-25 |             1 |             1 |            4
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |             1 |             2 |            3

(2 rows)
cqlsh> select * from k_master.fixture;

 hometeamid                           | datecreated | awayteamgoals | awayteamid                           | hometeamgoals
--------------------------------------+-------------+---------------+--------------------------------------+---------------
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |             1 | 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |             2

(1 rows)
cqlsh> select * from k_realtimeview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.333333 |                0.333333 |             0.333333 |            0.666667 |                 0.666667 |              0.666667 |            0.5
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |           0.342261 |                0.342261 |             0.339312 |            0.710923 |                 0.710923 |              0.697537 |       0.559794

(2 rows)
cqlsh> select * from k_batchview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.333333 |                0.333333 |             0.333333 |            0.666667 |                 0.666667 |              0.666667 |            0.5
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |           0.342261 |                0.342261 |             0.339312 |            0.710923 |                 0.710923 |              0.697537 |       0.559794
```

Second Run for Second Match (on speed layer)



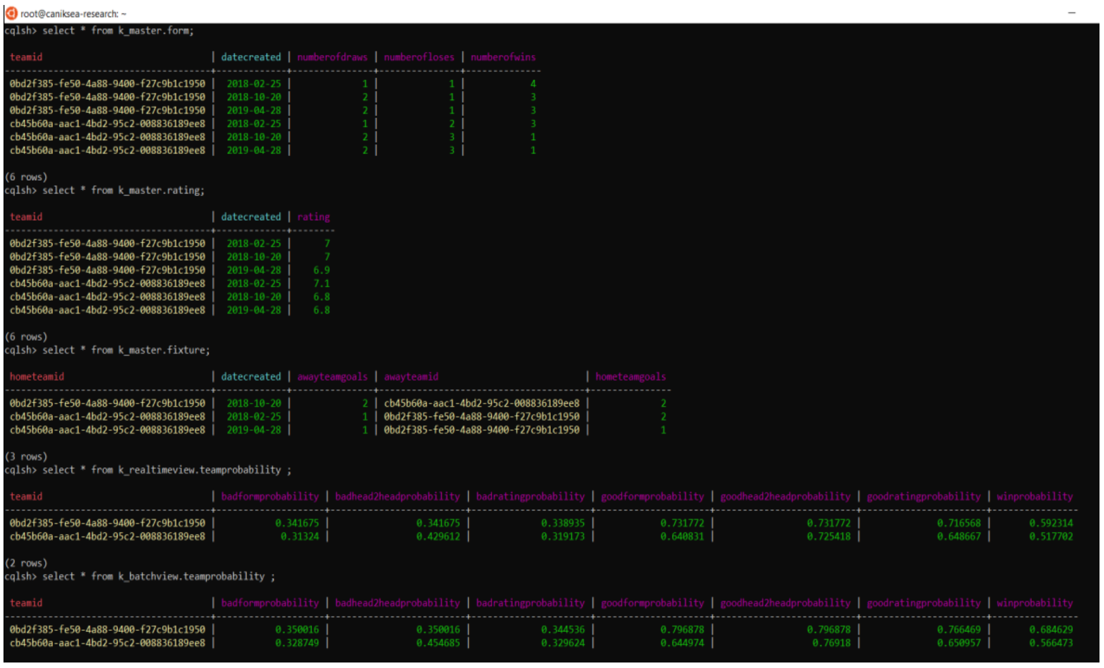
```
 teamid                               | datecreated | rating
--------------------------------------+-------------+--------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-02-25 |      7
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-10-20 |      7
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |    7.1
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-10-20 |    6.8

(4 rows)
cqlsh> select * from k_master.form;

 teamid                               | datecreated | numberofdraws | numberofloses | numberofwins
--------------------------------------+-------------+---------------+---------------+--------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-02-25 |             1 |             1 |            4
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-10-20 |             2 |             1 |            3
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |             1 |             2 |            3
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-10-20 |             2 |             3 |            1

(4 rows)
cqlsh> select * from k_master.fixture;

 hometeamid                           | datecreated | awayteamgoals | awayteamid                           | hometeamgoals
--------------------------------------+-------------+---------------+--------------------------------------+---------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-10-20 |             2 | cb45b60a-aac1-4bd2-95c2-008836189ee8 |             2
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |             1 | 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |             2

(2 rows)
cqlsh> select * from k_batchview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.333333 |                0.333333 |             0.333333 |            0.666667 |                 0.666667 |              0.666667 |            0.5
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |           0.342261 |                0.342261 |             0.339312 |            0.710923 |                 0.710923 |              0.697537 |       0.559794

(2 rows)
cqlsh> select * from k_realtimeview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.342122 |                0.342122 |             0.339218 |            0.710231 |                 0.710231 |              0.697054 |        0.55886
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |           0.333473 |                0.333473 |             0.333427 |            0.667358 |                 0.667358 |              0.667149 |       0.500934
```

Second Run for Second Match (on batch layer)



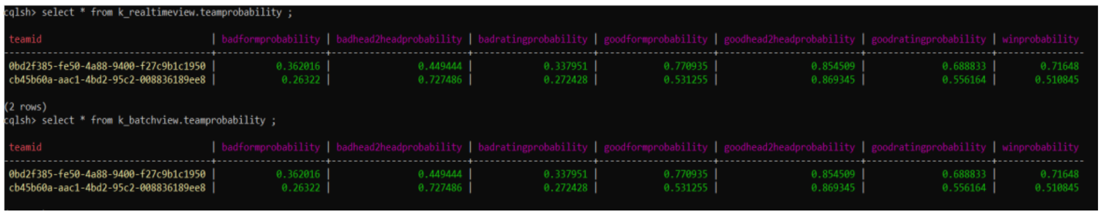
```
cqlsh> select * from k_realtimeview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.350016 |                0.350016 |             0.344536 |            0.796878 |                 0.796878 |              0.766469 |       0.684629
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |           0.328749 |                0.454685 |             0.329624 |            0.644974 |                  0.76918 |              0.650957 |       0.566473

(2 rows)
cqlsh> select * from k_batchview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.350016 |                0.350016 |             0.344536 |            0.796878 |                 0.796878 |              0.766469 |       0.684629
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |           0.328749 |                0.454685 |             0.329624 |            0.644974 |                  0.76918 |              0.650957 |       0.566473
```

Third Run for Third Match (on speed layer)



```
root@caniksea-research: ~
cqlsh> select * from k_master.form;

 teamid                               | datecreated | numberofdraws | numberofloses | numberofwins
--------------------------------------+-------------+---------------+---------------+--------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-02-25 |             1 |             1 |            4
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-10-20 |             2 |             1 |            3
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2019-04-28 |             2 |             1 |            3
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |             1 |             2 |            3
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-10-20 |             2 |             3 |            1
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2019-04-28 |             2 |             3 |            1

(6 rows)
cqlsh> select * from k_master.rating;

 teamid                               | datecreated | rating
--------------------------------------+-------------+--------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-02-25 |      7
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-10-20 |      7
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2019-04-28 |    6.9
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |    7.1
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-10-20 |    6.8
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2019-04-28 |    6.8

(6 rows)
cqlsh> select * from k_master.fixture;

 hometeamid                           | datecreated | awayteamgoals | awayteamid                           | hometeamgoals
--------------------------------------+-------------+---------------+--------------------------------------+---------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |  2018-10-20 |             2 | cb45b60a-aac1-4bd2-95c2-008836189ee8 |             2
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2018-02-25 |             1 | 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |             2
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |  2019-04-28 |             1 | 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |             1

(3 rows)
cqlsh> select * from k_realtimeview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.341675 |                0.341675 |             0.338935 |            0.731772 |                 0.731772 |              0.716568 |       0.592314
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |            0.31324 |                0.429612 |             0.319173 |            0.640831 |                 0.725418 |              0.648667 |       0.517702

(2 rows)
cqlsh> select * from k_batchview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.350016 |                0.350016 |             0.344536 |            0.796878 |                 0.796878 |              0.766469 |       0.684629
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |           0.328749 |                0.454685 |             0.329624 |            0.644974 |                  0.76918 |              0.650957 |       0.566473
```

Third Run for Third Match (on batch layer)



```
cqlsh> select * from k_realtimeview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.362016 |                0.449444 |             0.337951 |            0.770935 |                 0.854509 |              0.688833 |        0.71648
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |            0.26322 |                0.727486 |             0.272428 |            0.531255 |                 0.869345 |              0.556164 |       0.510845

(2 rows)
cqlsh> select * from k_batchview.teamprobability ;

 teamid                               | badformprobability | badhead2headprobability | badratingprobability | goodformprobability | goodhead2headprobability | goodratingprobability | winprobability
--------------------------------------+--------------------+-------------------------+----------------------+---------------------+--------------------------+-----------------------+----------------
 0bd2f385-fe50-4a88-9400-f27c9b1c1950 |           0.362016 |                0.449444 |             0.337951 |            0.770935 |                 0.854509 |              0.688833 |        0.71648
 cb45b60a-aac1-4bd2-95c2-008836189ee8 |            0.26322 |                0.727486 |             0.272428 |            0.531255 |                 0.869345 |              0.556164 |       0.510845
```