A FOUNDATION FOR DATA SCIENCE EDUCATION FOR INFORMATION TECHNOLOGY STUDENTS

by

RICHARD WAYNE PRICE

(Under the Direction of Lakshmish Ramaswamy)

ABSTRACT

With the explosion of data in today's environment, the ability to process and gain meaningful insights has become a critical skill. Observing several years of senior Information Technology students at Georgia Gwinnett College (GGC), we realized critical foundational knowledge was missing from these students' education making data science processing an extremely difficult task for them to master. GGC offers an Information Technology program rather than a Computer Science program. With the different curricula, many of the core Computer Science courses are not taught. We began evaluating different methods of filling these knowledge gaps to enable students to not only understand basic data science processing but also to gain the knowledge allowing them to move forward in this new and challenging career field.

The purpose of this research is to create a framework enabling Information Technology students to embrace this technology and succeed in Data Science related tasks. These students lack many of the core courses required in a Computer Science program but not in an Information Technology program which puts them at an extreme

disadvantage when learning data science technologies. These missing courses include operating systems, distributed systems, and remote communication methods. Most of the students in an Information Technology program have limited or no exposure to distributed file systems adding additional areas of missing critical knowledge. Prior to these students being able to successfully complete the normal day to day tasks for a data scientist, these core concepts must be taught and understood.

The contribution of this research is to identify core knowledge needed to remedy the knowledge gaps present in many Information Technology students and prepare them for data science concepts. We believe once these concepts are mastered, student performance will improve on data science tasks. This foundational material has been proven to remedy disparities in ability and confidence to execute these tasks regardless of gender and race.

This research is designed to build a computational foundation to enable Information Technology students to effectively utilize data science platforms including Hadoop, Spark or other data intensive tools. This dissertation does not address other pre-requisites for data science including statistical processing, visualization, and communication issues that must be addressed to develop a data scientist.

INDEX WORDS: Data Science Education, Distributed Systems, Information Technology Education, Operating Systems

A FOUNDATION FOR DATA SCIENCE EDUCATION FOR INFORMATION TECHNOLOGY STUDENT

by

RICHARD WAYNE PRICE

B.S., Millsaps College, 1976

M.S., George Mason University, 1995

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2019

A FOUNDATION FOR DATA SCIENCE EDUCATION FOR INFORMATION TECHNOLOGY STUDENTS

by

RICHARD WAYNE PRICE

Major Professor:	Lakshmish Ramaswamy
Committee:	Hamid Arabnia
	Liming Cai

Electronic Version Approved:

Ron Walcott
Interim Dean of the Graduate School
The University of Georgia
December 2019

DEDICATION

This dissertation is dedicated to my family. There is no way this could have been completed without your love and support.

During my journey to reach this point in my educational career, you have all been behind me and providing encouragement and support. When I was struggling and ready to give up, you supported and encouraged me to continue. Your confidence and assistance allowed me to persevere and complete this work.

Mom and Dad, thank you for instilling in me a strong work-ethic and perseverance to see a task through. Your example showed me how to commit to a task, work diligently and never lose sight of the truly important things in life. I wish you were here to celebrate with me.

To my daughters and son-in-law, thank you for helping and providing encouragement and, at times, the much-needed distraction to allow me to regain my perspective and motivation that helped me along the way.

To my in-laws, thank you for your support and willingness to allow me to be absent from many family gatherings while I pursued this goal. Your willingness to allow me to spend hours working and being absent from many gatherings greatly assisted me.

Finally, to my wife Elise. Thank you. You pushed me when I needed it, supported me when I was faltering, shared in my accomplishments, the disappointments and the frustrations along the way but you never let me get discouraged during this incredibly trying journey. Thank you for reading numerous documents to help produce the best possible product each time. There is no doubt, that without you, we would not be here.

ACKNOWLEDGEMENTS

First and foremost, I have to thank my advisor, Lakshmish Ramaswamy. You have been a friend, an advisor and an encourager for me during my pursuit of my Ph.D. You have walked through this journey every step of the way and provided invaluable guidance and insights that have let me successfully complete this momentous step in my academic career. Your guidance allowed me to thoughtfully consider the direction of my research and select the correct research projects and helped me correctly interpret the results of this work. I am honored to have been a Ph.D. student under your guidance. Your caring approach for your students has been a glowing example for how we should treat our students and has inspired me and provided me guidance in how to treat my own students.

Liming Cai, since our class together at the beginning of this journey, you have always been encouraging and helped me stay focused on the end goal. Your willingness to help me in every aspect, even something as simple as delivering paperwork for me has been a tremendous help. You have demonstrated the importance of teaching and assisting students in their pursuit of their educational goals.

Hamid Arabnia, we may not have always seen eye to eye on things but I have never doubted your motivation to make me the best possible educator/researcher possible. Your encouragement and guidance have helped me go beyond what I would have been able to accomplish without your example. Thank you for making an investment in my education and helping me to become a better researcher and teacher.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1

Introduction

1.1 Motivation

Today's students are faced with more and more difficult challenges especially when they begin dealing with large volumes of data. This problem is only becoming more critical as data continues to grow explosively. Current shortfalls in trained personnel exceeded 150,000 in August 2018 and the shortage is expected to increase [1]. With this shortfall, we began teaching Data Science basics to senior Information Technology (IT) students to help prepare them for challenges they might encounter in their careers. Shortly after we began teaching the data science introduction material, we discovered there were critical knowledge gaps that had to be filled before we could teach these concepts.

Based on this experience, we began pursuing a foundational framework to provide students the core computational knowledge required to be successful in acquiring basic data science skills. This dissertation does not address other requirements for data science including statistical processing, visualization and communication which must be mastered for a student to be efficient as a data scientist. While much of the missing computational knowledge is taught in a Computer Science (CS) program, it is missing from an IT program since the curriculum does not include many of these core CS courses. This dissertation is intended to address the knowledge gaps resulting in the difference between a Computer Science and Information Technology curriculum. The majority of

the students involved in these classes simply did not possess this foundational knowledge. This material was taught in the Software Development capstone course, where some of the most technical IT students are enrolled. This research aims to build a framework that can be leveraged to provide the core knowledge for IT students to prepare them for future data science courses and application.

The specific contribution of this dissertation is to identify core skills needed by Information Technology students before beginning data science tasks. Unlike a Computer Science curriculum where these core courses, Operating Systems and Distributed Systems, are part of the required courses, Information Technology curricula either do not have these courses or if they are offered, they are not required courses for many of the students.

When examining the educational requirements for a data scientist, Information Technology degrees are not listed for undergraduate graduates [2]. By providing the proposed foundational material, we believe students can complete the data science/analytics courses necessary for them to become productive employees in these fields.

1.2 Our Solution

The framework we built consists of the following major components:

- Basic operating system concepts and navigation,
- Distributed systems
- Remote communication and lookup methods, and
- Distributed file systems

We began by teaching basic operating system concepts. Unix was selected to provide hands-on experience in this environment as well as for the popularity of this operating system in distributed systems. Topics covered included:

- Basic naming concepts
- Symbolic links
- Unix file system
- Sudo and associated directory and administrative rights
- Command line navigation

After teaching basic operating systems, we then taught distributed system concepts including:

- Characteristics
- Transparency
- Consistency
- Middleware concepts
- Heterogeneity

We created and utilized a "distributed application" using Java to teach the heterogeneity of a distributed system. This was done since these students were familiar with Java and it could be used to demonstrate a heterogenous system running across multiple operating systems.

Once students understood the Unix operating system and Distributed Systems, this greatly simplified teaching distributed file systems. We chose the Hadoop Distributed File System (HDFS) for our students since it represented the different aspects of a distributed file system and showed the failure resilience that can be achieved with a

distributed system. Also because of the popularity of this distributed file system, we felt it was important for students to be exposed to this tool.

We hypothesized that the foundational framework would provide students with the core knowledge required to understand data science basics. This would allow them to perform routine data science processing tasks.

To test our hypothesis, we conducted attitudinal surveys of our students to help us identify those areas where they experienced difficulties in learning this material. Once we had identified the candidate material to assist students learning this material, we created a set of instructional material covering these concepts. We then created pre- and post-tests to measure the effectiveness of this material. An additional measure was a data science project that tested the students' understanding of data science techniques.

We were fortunate to have control and experimental sections to test our hypothesis (covered by GGC IRB 17043, Appendix A). We administered the pre-test to these sections and discovered no significant difference between the control and experimental sections. After teaching the foundational material to the experimental section, we then taught both sections the same data science concepts and assigned the same project. We administered the post-test to experimental and control sections. This allowed us to not only compare our observations of student ability but also to provide metrics for the effectiveness of this approach both from the post-test and project grade differences.

The research questions that we are addressing in this dissertation include:

- **RQ1 - Is there a set of material that will improve success in learning data science programming, data collection, and cleaning?** Based on our experience

teaching this material, we hypothesize that the foundational material proposed will provide better student success in basic data science tasks and understanding.

- **RQ2 - Is there a different set of material needed for minorities and/or women?** When confronted with these tasks, is a different set of foundational material required for women or minorities? Based on enrollment rates and industry percentages of minorities and women, we wanted to ensure that the foundational material was effective in preparing all students for these tasks.
- **RQ3 - Can this be formalized into a framework providing improved student understanding and performance in data science tasks?** We hypothesize that the material can be used as a framework to improve student understanding of data science concepts and improve performance on data science tasks.
- **RQ4 - How can we measure success on data science tasks?** We hypothesize that the pre- and post-tests can be used to ensure similar student preparation before presenting the foundational material and use the post-tests to evaluate the mastery of these concepts. In addition, we hypothesize that students exposed to this material will be more capable of performing these tasks in their industry career.
- **RQ5 - How do we address the differences in a traditional CS program and an IT program where many of the core subjects are not required?** We hypothesize that there are significant differences in the core knowledge between CS and IT students. Our hypothesis is that we can bridge these knowledge gaps by teaching the material selected in our foundational framework.

1.3 Contributions

A Foundation for Data Science Education for Information Technology Students results in several contributions for an Information Technology Data Science program. Many Data Science programs exist in various disciplines including Computer Science, Management Information Systems, and Statistics but a data science program in Information Technology is a rarity. Based on the focus of this discipline, different approaches are taken to complete a student's education. Information Technology is more focused on application and problem solving in industry while Computer Science is more focused on the theory underlying this technology.

First, this dissertation addresses shortcomings in student knowledge in areas including basic operating systems and different methods of utilizing operating systems. Second, the introduction of distributed system concepts prepares the student for introduction of distributed file systems, middleware, failure resilience, and concepts related to algorithm development and employment. Specifically, this work contributes to the following issues in the CS and IT educational communities:

- Remedying knowledge gaps and preparing students for data science concepts
- Elevating student performance in data science related tasks
- Increasing graduates with data science exposure
- Remedying any minority or gender disparities in ability and confidence to perform data science tasks and learning

1.4 Overview of Dissertation

- Chapter 2 provides a history of data science and literature review.

- Chapter 3 provides a plan for inclusion of data science in an Information Technology program in the currently existing courses. This chapter also highlights critical differences between IT and CS curriculum exposing core knowledge not taught in the traditional IT program.
- Chapter 4 provides information from industry on the expected duties and skills required for a new graduate in a data analyst or data scientist role. These areas vary widely with different roles and methods used for data analysis. The intent of this dissertation is to provide a set of core skills that allow the graduate to be prepared for these roles.
- Chapter 5 describes Efficient Processing of Range Queries Over Distributed Relation Databases. This material is used to introduce the heterogeneity of distributed systems to students using a familiar technology.
- Chapter 6 introduces A Framework for Introduction to Data Science for Information Technology Students.
- Chapter 7 presents and interprets the conclusions and details the framework created to support Information Technology students.

CHAPTER 2

Data Science History

Data Science is not a new endeavor. In 1961, John W. Tukey published a paper entitled "The Future of Data Analysis" [3] While Tukey was not certain what the field would entail, he recognized a new emerging field whose subject of interest was in learning from data and data analytics. Over the next ten to twenty years, three more statisticians began calling for their colleagues to go beyond the normal limits of theoretical statistics. Chambers urged his colleagues to examine methods of data preparation and presentation rather than pure statistical modeling. Breiman stressed prediction rather than inference. Cleveland suggested the name of Data Science for this new field of study. [4]

Tukey was a statistician and his view of Data Science was to reformulate statistics and to create a branch that would cover the emerging predictive analysis that he believed was going to become necessary. He identified four driving forces that were emerging that propelled data science forward. It is amazing how similar these factors were to those that exist today. First, Tukey wanted to continue to use the formal theories of statistics. He wanted to couple these with developments in computing and display devices to allow better predictive analysis. He also recognized the rapid growth in data in many fields where larger bodies of data needed to be processed and analyzed. Finally, he encouraged application of "data science" technologies in more disciplines and domains. He

recognized the necessity of focusing the analysis in the domain rather than in a statistical hierarchy.

Following its uncertain start as a renamed statistics discipline, Data Science has continued to be refined, changing from simple statistical analysis to become a new discipline leveraging statistics, computer science and domain knowledge in a particular area. This transition occurred over many years and we will describe some of the highlights of this journey. Over the years, there has been confusion with naming this emerging field. It has been called Big Data, Data-Intensive Computing, and Data Analytics, just to name a few. We will be discussing the history and challenges associated with these under the history and challenges of Data Science.

The exponential data growth we see today had its beginnings many years ago, 1941, when the term "information explosion" was first used. This growth in data was validated in the 40's by men such as Fremont Rider who estimated that the American university libraries were doubling in size every sixteen years. This trend not only continued but accelerated leading Derek Price to estimate scientific knowledge was increasing by a factor of 10 every half-century. One of the major early concerns was storage requirements. This led many to advocate for keeping all information requirements to a minimum. [5]

In 1974, Peter Naur in his book Concise Survey of Computer Methods used the term Data Science defining it as "The science of dealing with data, once they have been established, while the relation of the data to what they represent is delegated to other fields and sciences." [6] Naur was concerned with the processing of already acquired data

that had been cleaned, formatted and put in a form that allowed for easy manipulation. He was relying on "other fields" to provide useful data for analysis. [7]

In 1977, the ISAC, International Association for Statistical Computing, was formed with the stated mission to "link traditional statistical methodology, modern computer technology and the knowledge of domain experts in order to convert data into information and knowledge." [7] In our opinion, this is the true beginning of Data Science. While others had stated the need earlier including Tukey, this was the first body that formalized the need to fuse statistics, computer science and domain expertise.

I.A. Tjomsland stated in 1980, "Data expands to fill the space available." In 1983, the exponential growth in data had continued and Americans could receive via the media 8.9% more words than the previous year. [5] 1989 saw the start of the Knowledge Discovery in Databases conference which would mature into the ACM SIGKDD conference in 1995. In 1990, Peter J. Denning stated "The rate and volume of information flow overwhelms our networks, storage devices and retrieval systems. It is beyond the human capacity for comprehension." [5]

Business Week ran a cover story in 1994 revealing that companies had started gathering large amounts of personal information with the intent of developing new, targeted marketing campaigns by leveraging this data. This was followed in 1999 by an article written by Jacob Zahavi in which he pointed to the massive amounts of data and stating "Scalability is a huge issue in data mining." Zahavi clarified the Data Science problem even more by stating, "Conventional statistics methods work well with small data sets. Today's databases, however, can involve millions of rows and scores of columns of data." [7] Zahavi stated the problem as he saw it in 1994 without realizing the

growth of data. He limited his observations to the issues at hand without the future prediction of the massive data that would be produced in the next few years. He did identify that we had to switch to a different analysis model and leverage computational power to be able to analyze the data that was being generated.

In 1992 a new programming language began development to be released in 1995. This language, R, was conceived to handle processing of statistical data and provided graphical tools for easier visualization. Since its introduction, it has become one of the languages of choice for processing statistical data. It is an integrated suite of software providing the ability to manipulate large data sets, a set of tools for data analysis and graphical features for presentation of the data analysis results. [8]

In 1997 the first use of the term "Big Data" was made in an ACM article. This article addressed the massive data sets and the processing of them. Challenges specifically identified at this point were memory, local disk space, remote disk space and access time and finally, the problem of presenting the data. This article discussed methods of data visualization and achieving acceptable performance with these large data sets. [9]

Bryson et al [10] were the first to use the term Big Data in a Communications of the ACM Data in their article "Visually Exploring Gigabyte Data Sets in Real Time." In this article, the authors identified visualization and modeling techniques to allow for meaningful, real-time presentation of data. They specified a simple three-step pipeline for visualization: 1) The visualization is specified, 2) The visualization graphical geometry is computed accessing relevant data and 3) Finally, the geometry is rendered. In addition,

they discussed rendering techniques such as sparse traversal, using only a subset of the data and compression, which can be either lossy or lossless depending on the field.

Data continued to grow and in 1999 estimates were the world produced 1.5 exabytes, about 250 megabytes for every man, woman and child on the planet. Contrast this with 2006 where the estimates were that 161 exabytes were created in that year alone with estimates that this would double every 28 months. By 2011, the world information storage capacity was growing at annual rate of 25%. [5] With the massive increase in data, new tools and techniques had to be developed to allow us to turn this into useful information.

In 2002 the Data Science Journal was launched which dealt with the management of data and databases. In 2003, the Journal of Data Science was launched. In contrast with the Data Science Journal, this journal dealt with almost everything that has something to do with data: collecting, analyzing, modeling and most importantly, its application. [11] We believe this was a significant change in the mindset at that time. In a period of one year, a new journal was introduced that dealt with the complexity surrounding Data Science and at that time Big Data.

Ben Fry [12] published his dissertation in 2004 calling for a new field of study that combines computer science, statistics, data mining, information visualization and human-computer interaction. Fry continues showing visualization as one of the major areas in data science.

Hadoop 0.1.0 was released in 2006 and the term NoSQL was reintroduced in 2009. Both of these point to the nature of data being analyzed using Data Science; largely unstructured or semi-structured at best. This was a big move away from the data

processing at the time which was largely relational based allowing for much simpler mapping and processing. [7]

Also, in 2006, Apache Pig began at Yahoo and was moved into the Apache library in 2007. This tool was created to accommodate the need for multi-step extraction, transformation and load (ETL) processing inside Hadoop. This provides the ability to split processing into multiple steps that are optimized using MapReduce jobs. It has since been expanded to include Pig-on-Spark [13]. NumPy, a Python library targeted at scientific processing was released allowing for easier numerical processing [13].

Python Pandas was released in 2008. This library provides data structures and tools using the Python language to facilitate data analytics tasks. This library provides data munging and preparation functions as well as tools to assist in data analysis and modeling. Statistical models included in Pandas are linear and panel regression [14].

In 2008, Bryant et al [15] published "Big-Data Computing: Creating revolutionary breakthroughs in commerce, science and society." In this article they addressed Walmart storing 4 petabytes of data and applying machine learning to detect patterns effecting the business. This included data such as the effectiveness of the supply chain, pricing strategies, advertising campaigns and inventory management. Several challenges were identified at the time. These included high-speed networks; 1 terabyte of data took a full day to transfer at that time. Other challenges included cluster computer programming, lack of cloud computing, machine learning and data analysis techniques, security and privacy. Many of the challenges described still apply today although some are simply because the size of the data has increased proportionately with the increases in technology.

In 2009, Data Science was being paid a tremendous amount of attention and people with these skills were highly sought. 2009 saw articles published about what skills a data scientist should have; statistics, data munging which is the cleaning, parsing, proofing and extraction of data from various sources; and finally, the ability to communicate the analysis of the data through story-telling, visualizations and the ability to visually advocate a hypothesis. [16]

Also, in 2009, U.C. Berkley began working on what has become one of the most preeminent data science tools today, Apache Spark. Spark is an in-memory processing engine providing better semantics and stream processing allowing more frequent computations of data science tasks. The initial Berkley project has been supplemented by Apache to include Spark SQL, MLib and improved stream processing. These additions have allowed Spark to continue to grow and become one of the foremost data science tools. Spark introduced the concepts of Resilient Distributed Datasets to allow for better failure resilience and speed of processing since these objects are often stored in memory rather than on disk. Today, Spark has grown to be the most widely used data science tool. [2] Spark continues to grow and mature today providing support for Scala and Python for pre-processing data.

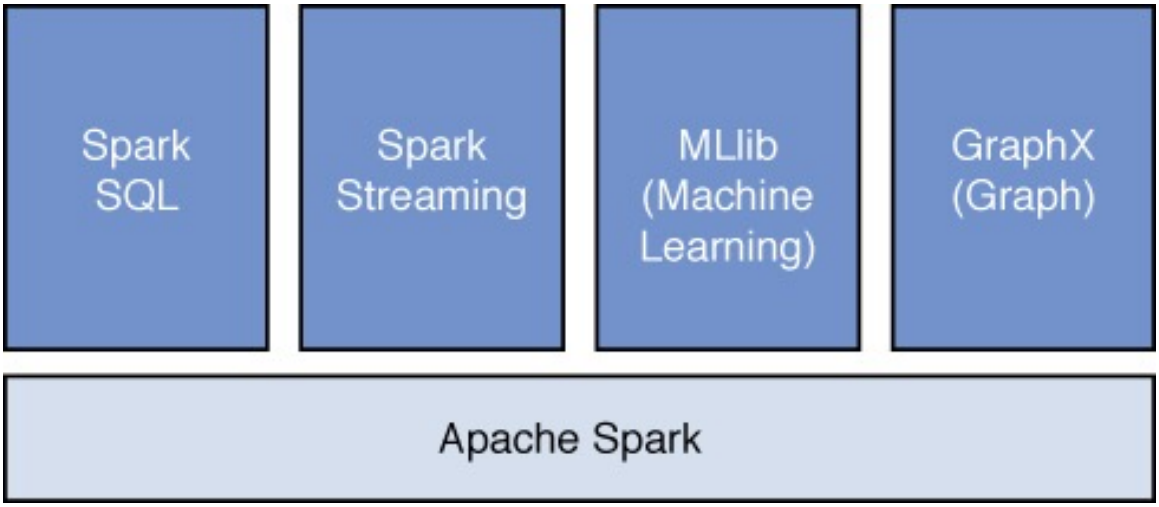

Figure 2.1: Apache Spark

Spark is made up of multiple components as shown in Figure 2.1[13]. The Spark SQL engine provides an alternative to Hive for SQL on a distributed database. The Spark Streaming engine allows processing of streaming data utilizing Spark. Spark MLib is a collection of machine learning libraries that are integrated with Spark and provide machine learning algorithms for use with data science processing. GraphX provides a library of visualization tools allowing the data scientist to more easily present their findings.

2010 saw the announcement of "a new kind of professional… the data scientist." The goal of a data scientist is to search for hidden nuggets of gold buried in the mountains of data. This professional combined the skills of software programmer, statistician and storyteller/artist to extract the meaning of the data. [17] In 2011, job postings for data scientists jumped by an incredible 15,000%. [7]

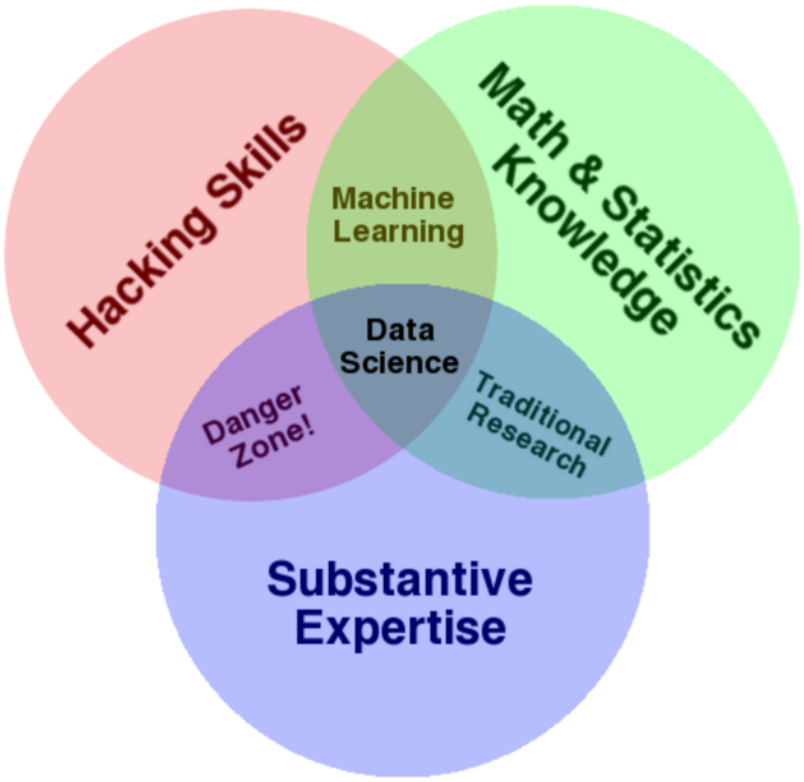

Figure 1.2: Data Science Venn Diagram

In Figure 2.2 [18], Conway represented the expertise needed in 2010 to be successful in data science. He showed data science needed a blend of talents in computer science, math and statistics and the domain being analyzed. He showed areas where one of the skills was missing and showed how they were not truly data science. A combination of CS and Statistics will probably lead to machine learning algorithms to support decisions. CS and domain experience with no statistical knowledge may lead to erroneous data collection and interpretation. Statistics and domain knowledge may lead to simply traditional research. Only in combination of all three areas can data science flourish. Conway made one more point that is very telling. At the time he published this,

he stated regarding data science there is an "utter lack of agreement on what a curriculum on this subject would look like."

2010 also saw the start of a new tool, Apache Flink. This was adopted into the Apache group in 2014 and provides improved stream processing. Flink has a very similar architecture to Spark but provides for improved iterative processing and true stream processing instead of the batch type processing implemented in Spark. [19]

Facebook saw the need to create a new messaging platform in 2010 and leveraged Apache HBase to provide this system allowing them to process massive amounts of data using natural language processing. HBase allowed the query of billions of rows containing millions of columns of data. HBase was modeled on Google's Bigtable. This system allowed distributed data storage and provided data resilience and ease of querying with built in API's [20]. Facebook also began developing a tool this year, Hive, that is now part of the Apache suite. This tool provides for more conventional data querying of data stored in the HDFS. This tool allows SQL syntax querying on this distributed data [21]. Since its introduction, this has been extended to include Hive-on-Spark.

In 2011 in a book entitled What is Data Science? [22], Loukides presented several interesting points. First, his definition of a data application as one that "acquires its value from the data itself and creates more data as a result", moved the focus away from the data and onto the analysis which produced useful intelligence to solve the problem. Next, he enumerated steps necessary for data analysis including data conditioning, cleaning, formatting, and examining for patterns. Once the data is suitable for processing, you can examine the data for the quality of the data. He recommended leveraging machine learning to process the data and gave several tools that could be useful including PyBrain,

Elefant, Weka and Mahout which coupled with Hadoop. He then recommended building statistical models and finally visualizing the data for interpretation. Again, he recommended tools including GnuPlot and Processing. Processing was created by Ben Fry whom we discussed earlier when working with visualization.

In 2012, Boyd et al [23] in "Critical Questions for Big Data: Provocations for a Cultural, Technological, and Scholarly Phenomenon", presented many interesting points. First, they called into question the term Big Data by addressing the fact of technology growth stating "it has been used to describe data sets large enough to require supercomputers, but what once required such machines can now be analyzed on desktop computers." Their conclusion from this is that "big" is less about the size but rather about the ability to search, aggregate and analyze large data sets. They proposed a new definition of Big Data based on three principles. 1) Technology – maximize computing power to efficiently run algorithms to gather, analyze, link and compare large data sets. 2) Analysis – identify patterns in order to make claims in the problem space. 3) Mythology – the belief that large data offers a higher form of intelligence and knowledge than can be generated using traditional data and analysis tools. Several challenges were identified: 1) Data accuracy – how do you handle errors in the data. 2) Apophenia – seeing patterns where they don't exist simply because the enormous data set points to that conclusion. This caution was to keep users from utilizing standard statistical techniques such as p-value. 3) Incomplete data – they recommend "whole" data not big data. For example, Twitter doesn't represent everyone and even those with an account do not necessarily share every tweet. 4) Ethics in data collection – just because you can get the data doesn't mean that it is ethical to collect, analyze and report on this data.

Another data science tool that was introduced in 2012 is Apache Sqoop. This tool allows for transferring data from Hadoop to structured storage such as a relational database. Sqoop slices datasets and uses a map-only job to transfer data in each partition to the storage system. [24] This same year, Apache Flume began making its presence known. Flume allows for efficient collection, aggregation and transfer of large amounts of log data to facilitate processing this information. Flume retrieves data and then stores it in the HDFS. [25]

Data continued to be collected at exponential rates. In 2013 IBM shared a statistic that 90% of the data in the world has been created in the last two years. [7] While the large data volumes contributed significantly to the difficulty of the analysis problem, the shift from structured to unstructured data required the development of new tools and techniques.

This same year, Chen et al [26] identified key challenges to the Big Data space in their article "Big data challenge: A data management perspective." In this they identified three factors that made data big. These were volume, velocity and variety. They list as major challenges big data diversity, big data reduction, big data integration and cleaning, big data indexing and querying and finally big data analysis and mining. They developed a workflow as shown in Figure 2.3. This workflow did allow for capture of data from multiple sources. The first step in their workflow was what they called data integration and cleaning. They suggested the use of crowd sourcing to integrate and clean the data. This step was to validate the data. Next, they had a data reduction step to allow them to reduce the amount of data to allow traditional analysis methods to be used. One of their suggestions was to use machine learning to reduce the data. Querying was to be done

using traditional queries, possibly leveraging a distributed system to allow for more expedient processing. Finally, they had an analysis and mining step which they stated was “an important challenge. Data is increasing with an exponential result.” No real guidance was offered here nor was visualization discussed as a method of communicating their findings.

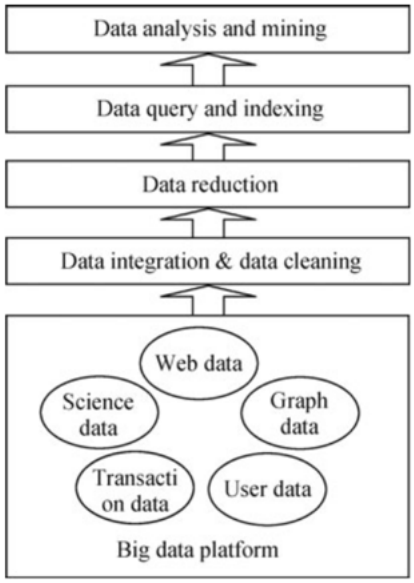

Figure 2.2: Big Data Workflow process as developed by Chen et al

Guo et al [27], published an article “Data Science Workflow: Overview and Challenges”. In this article they identified four main phases for data science processing. First, was the preparation of the data. In this step the data scientist needs to acquire the data, determine the source and deal with the different ways of data delivery: files, streamed, automatically generated, or manually entered. Data then needed to be reformatted and cleaned. This step converts the raw data into a convenient format for a programmer to run an analysis. If there are semantic errors, missing entries, inconsistent formatting or any other type of error, these need to be corrected in this step. The second

step is running the analysis, write and execute the computer programs to analyze the data and obtain insights it contains. The third step is to reflect on the results and interpret the output. This may require going back to the analysis phase. The data scientist needs to determine how to communicate the insights obtained. Finally, the result needs to be disseminated as actionable intelligence.

In 2014, George et al [28] pointed to several events that were driving the Big Data/Data Science innovations at that time. First the UN had established Global Pulse which was an initiative that used data sources such as mobile phones, mobile payments and other real-time data sources to collect, mine and analyze the data to detect emerging vulnerabilities in developing countries. They recognized that Big Data was being generated through multiple sources such as internet clicks, transactions, user-generated content and social media. In this article, they changed the definition of "big" from the pure size to how 'smart' it is. This changed the definition of big from size to the kind of information that can be gleaned from the data. One of the key concepts they proposed was that micro data would provide a rich map of individual behaviors and actions rather than simply focusing on the success or failure of a problem. We believe this was one of the first migrations to using this "micro data" to predict individual behavior whether that is from a buyer, a sensor or an internet click collection. This data can be mined to provide meaningful insights into the individual contributor. Also, George et al recognized the need for data sharing agreements as well as mechanisms for data protection and privacy. Their techniques relied more on Bayesian statistics and stepwise refinement models rather than traditional p-value comparison. Finally, they recognized consilience and how evidence from multiple, independent, unrelated sources can lead to strong conclusions.

In 2017, Cao [29] wrote an article "Data Science: Challenges and Directions". In this article, Cao describes the requirements and challenges for a data scientist. According to Cao, data science required systematic thinking, solid methodologies and the reliance on machine learning to create machine intelligence with regard to the data. Data science problems are complex systems requiring the scientist to represent, learn, simulate, reinforce and transfer human-like intuition, imagination and creative thinking through human-data interaction and cooperation. The goal is to translate data into insights and intelligence to assist in decision making. Data science can be summed up in his equation, "data science = {statistics ∩ informatics ∩ computing ∩ communication ∩ sociology ∩ management | data ∩ domain ∩ thinking}". The | he defines as "conditional on". Our interpretation of this is that data science is the intersection of statistics, informatics, computing, communication, sociology, and management. Unless we think and understand the data, the domain and have a defined problem solution for the data and insights we are trying to analyze, we are doomed to failure. Data scientists must understand many domains. If they are unable to formulate problems relevant to the domain and think through the types of information they are trying to analyze and recognize solutions when presented, they will not be successful in this field. Figure 2.4 shows the relation of these critical areas to form the core knowledge and skills for a data scientist.

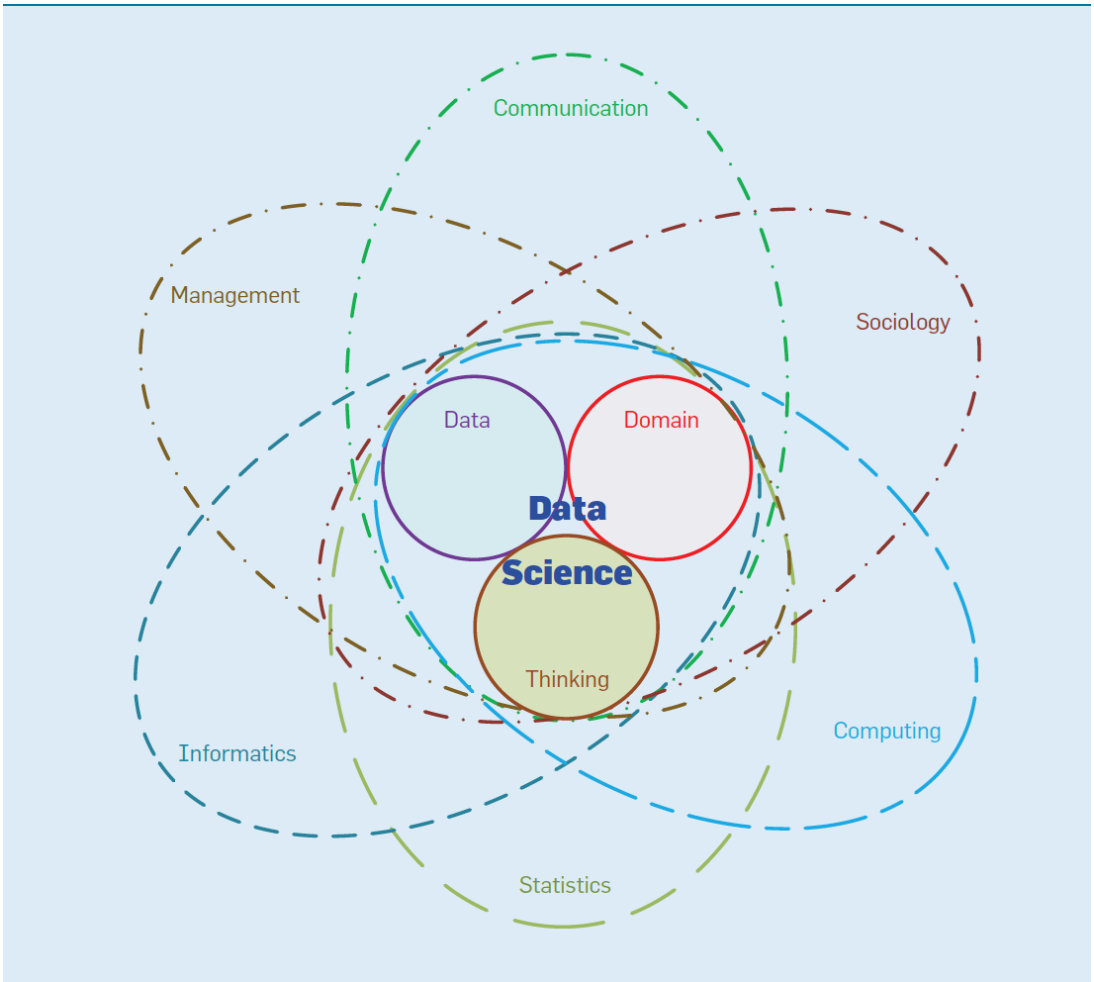

Figure 2.3: Data Science core competencies

Data scientists search for hidden relationships in data and use these relationships to solve critical business problems in complex domains. This data contains all types of information: social, transactional, sensor, web clicks and many more that require the scientist to perform analysis on the data and gather insights.

Data science was selected as one of the top eight fields in 2017. [30] There are many opportunities and the demand is extremely high for qualified applicants. There are many challenges to overcome but this promises to be an exciting and rewarding career for the foreseeable future.

In 2017, ACM approved the Data Scalability Goals for an Information Technology program. These included the following goals:

1. Perspectives and impact
    a. Discuss emerging field of data science
    b. Identify sources of large volumes of data
    c. Recognize challenges in analytics of very large volumes of data
    d. Describe how analytics can be used in major functional areas of an organization
2. Large-scale data challenges
    a. Define and describe large-scale data challenges of volume, variety, velocity, and veracity
    b. Define and describe challenges of large-scale data analytics in diverse sectors such as sensor networks, finance, retail, genomics, and social media
    c. Compare different data platforms that can be used for processing and generating large-data sets
    d. Use a statistical programming language such as R or Python
3. Data management
    a. Discuss common Extract Transform Load scenarios
    b. Apply data preprocessing techniques – data integration, data cleansing, data transformation, and data reduction
    c. Discuss how to extract knowledge and insights from large and complex collections of digital data
    d. Use data mining software to perform data mining

4. Methods, tools and techniques
    a. Explain technical foundations of the commonly used data analysis methods
    b. Apply appropriate data analysis methods to solve real-world problems
    c. Use tools such as R and RStudio, MapReduce/Hadoop and SAS
    d. Communicate the results of data analysis to technical and management audience
    e. Effectively communicate the results of data analysis using visualization
5. Data governance
    a. Identify the importance of data governance for managing large-scale data
    b. Identify logical and physical access security controls to protect data
    c. Identify current social, ethical, legal, and policy issues caused by the large-scale data analytics
    d. Define data ethics
    e. List regulatory compliance rules and regulations applicable to data management
6. Application
    a. Define an organizational problem as an analytics problem
    b. Describe how to best apply large-scale analytics methods and techniques in addressing strategic organizational problems
    c. Apply a data analytics lifecycle to a case study scenario
    d. Implement data-intensive computations on cluster and cloud infrastructures.
    e. Examine the impact of large-scale data analytics on organizational performance using case studies [31]

Survey data science programs in Computer Science, Statistics or Business schools as updated in 2018 and 2019, these programs recommend a combination of the following coursework:

- Programming
- Introduction to Unix
- Databases and data structures
- Algorithm design and analysis
- Data security
- Machine learning/AI
- Data analytics and visualization
- Applied statistics
- Bayesian analysis and statistical decision making
- Data management for data science
- Data privacy and security
- Linear algebra
- Data mining [32] [33] [34] [35]

De Veaux et al published their "Curriculum Guidelines for Undergraduate Programs in Data Science" [36] where they identified several critical aspects for Data Science education. First, data science is an interdisciplinary area. A data scientist has to not only have computational skills but also requires domain and statistical/mathematical knowledge. They identified the recursive nature of obtaining data, data identification, data collection, curation, munging, and processing. Interestingly, they identified two different cultures in data science; one being algorithmic, computational expertise, and the

other being data models, statistical. Data science offers the opportunity of combining these to solve problems. Their opinion of data science is that is consists of problem-solving approaches melding statistics, computer science and mathematics. We believe that this must also include domain knowledge to help the data scientist interpret data and determine the correct data to guide the analysis.

According to the "Big Data: Business, Technology, Education, and Science Ubiquity Symposium", published in 2018, data scientists require not only technical skills and knowledge, but also soft skills including personal drive, the ability to learn quickly in a self-directed way and, above all, interpersonal skills and teamwork. [37] Also in 2018, Carmichael et al published an article "Data science vs. statistics: two cultures?" [38] in which they state that data science is defined as the intersection of math and statistics, computing and knowledge of a particular domain. Implicit in this definition is a focus on solving problems. Carmichael et al expanded this definition to include:

- Data gathering, preparation and exploration
- Data representation and transformation
- Computing with data
- Data modeling
- Data visualization and presentation
- Science about data science

By defining data science in this manner, the authors felt they gained improved understanding of how data scientists spend their time and effort as well as putting more focus on the value each tool provides insights into the data.

In 2019, IT Career Finder published their definition of a data scientist in today's environment. Included in this definition is the requirement for a data scientist to possess analytical skills, technical prowess, and business acumen to effectively analyze massive data sets and think critically to solve problems. [2] The day to day responsibilities identified include:

- Perform data-mining, modeling
- Stay current with emerging tools and techniques including those in machine learning and statistical modeling
- Exercise excellent oral and written communication
- Develop customized algorithms to solve problems
- Use data visualizations to communicate
- Use Hadoop to analyze and mine data sets
- Have computer programming skills
- Work in a team setting

We believe the components of Data Science include all of those shown in Figure 2.5. Data Science is a composite of many different skills including computer processing and programming, data acquisition and cleaning, statistical analysis, interpretation of results and communication. For the data scientist to be successful, they need to not only understand the technical aspects of programming, machine learning and distributed systems but also need to understand the domain they are serving. They need to understand the data that is being processed; what kind of data, how that data can be cleaned and processed, how it can be stored, how to make meaningful interpretations of this data, and what the ethical implications of using this data are. The data scientist needs

a grounding in statistical methods to be able to assist in determining the significance of findings but also needs to be able to identify those non-statistically significant findings that can lead to the treasure buried within the data. Finally, the data scientist must know how to present key findings from this sea of information, how to present these findings in the most efficient manner to allow rapid use of key findings.

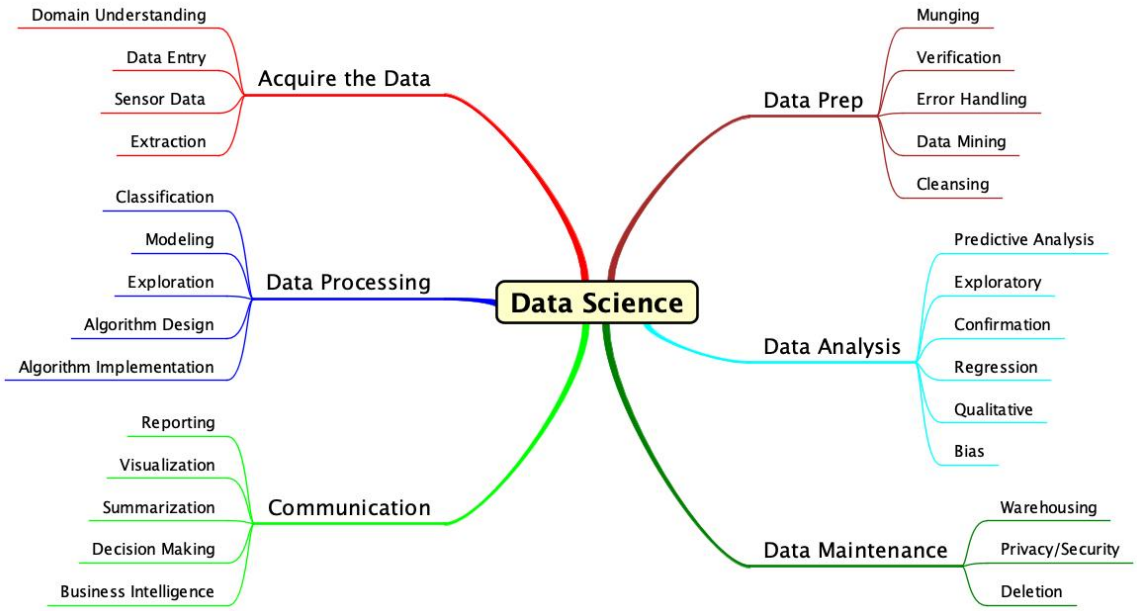

Figure 2.4: Components of Data Science

CHAPTER 3

Key Differences in CS and IT Programs

Before determining how a data science program will differ from traditional CS and statistics programs, we feel we first need to discuss the current data science programs. To begin this discussion, we examined the courses and outcomes from multiple data science programs. These programs are typically in one of three schools: Business, Statistics, or Computer Science departments. The majority of the programs we were able to find are associated with Business or Statistics. We did not find any data science programs in Information Technology programs.

Of the data science, analytics or related programs we reviewed, about 41% were statistics based, another 23% were in the business school and the remaining 36% of the programs were either in the computer science school or in a collaboration between the computer science and statistics departments.

Business School programs: [39] [40] [41] [42] [43] [44] [45] [46] [47]

Objectives include:

1. Use predictive modeling, communication and quantitative reasoning to solve business problems.
2. Learn to manage and maintain the hardware and software systems businesses use to collect, organize, and secure data.
3. Learn to manipulate and evaluate data to inform strategic business decisions and process improvements.

4. Understand how to work with the ever-increasing amount of data to help businesses succeed by improving operational efficiency and risk calculation.
5. Assess the challenges and opportunities presented by massive unstructured data.
6. Implement a variety of techniques and technologies for analyzing massive data sets.
7. Interpret, analyze and validate the results of big data analysis.
8. Apply appropriate data warehousing and database management solutions to meet the security, quality, storage and privacy needs of organizations.
9. Use data analytics to examine raw data using analytics that focus on inference through the transformation of data to actionable information that improves decision-making.

Coursework includes courses that cover the following material:

1. Managing and governing, extracting, merging and preparing large data sets.
2. Linear models including multiple linear regression and model building in business decision making and applications.
3. Business process analysis.
4. Business for decision making, statistical decision making.
5. Visual effects.
6. Data mining.
7. Decision support presentation.
8. Data engineering.
9. Hacking.
10. Social Media and Content Marketing.

Statistics School programs: [48] [49] [50] [51] [52]

Objectives include:

1. Think logically and analyze information critically.
2. Visualize, present and communicate analytical results.
3. Analyze complex systems and solve real-world problems through the analysis of data, particularly very large data sets.
4. Apply a mathematical approach to the analysis of data.

Coursework includes courses that cover the following material:

1. Introduction to data science using R or methods in data science.
2. Basic programming, data structures and algorithms.
3. Databases.
4. Calculus, linear algebra, probability, application of statistics, and linear algebra.
5. Efficient computation and Big Data.
6. Time series analysis.
7. Programming, algorithms, data structures.
8. Data mining.

Computer Science programs: [35] [32] [33] [34] [53] [54] [55] [56]

Objectives include:

1. Apply statistical methods and procedures to model a problem. Collect and analyze data to evaluate the solution.
2. Assist organizations with the effective use of data.
3. Apply ethics, interdisciplinary research, and communications skills specific to data analytics, practiced in a variety of application domains.

4. Ability to apply computer science principles relating to data representation, retrieval, programming, and analysis.
5. Ability to apply mathematical and statistical models and concepts to detect patterns in data, and to draw inferences and conclusions supported by data.
6. Ability to design, implement and analyze the software that manages the volume, heterogeneity and dynamic characteristics of large data sets and that leverages the computational power of multicore hardware.

Coursework includes courses that cover the following material:

1. Programming, Introduction to Unix, database, data structures.
2. Cloud implementation strategies.
3. Algorithm design and analysis.
4. Data Security.
5. Machine Learning/AI.
6. Data Analytics and Visualization.
7. Principles of informatics.
8. Applied statistics.
9. Bayesian Analysis and Statistical Decision Making.
10. Data management for data science.
11. Data privacy and security.
12. Linear algebra.
13. Data mining.

Many of these programs require specializations in other areas such as bioinformatics, finance, marketing, big data analytics, cloud virtualization, health informatics, and data analysis. Many of them, like Ohio State's program [54] are multi-disciplinary using faculty from Arts and Sciences, Engineering, Medicine, and Business.

Data Science is truly a multi-discipline program utilizing expertise in computer science, statistics and math, and a subject matter field. This is illustrated in Figure 3.1 [46] from Massey University showing their concept of the formulation of a data science program. This is echoed in many of the programs that exist today. None of the programs reviewed were focused solely in a single discipline.

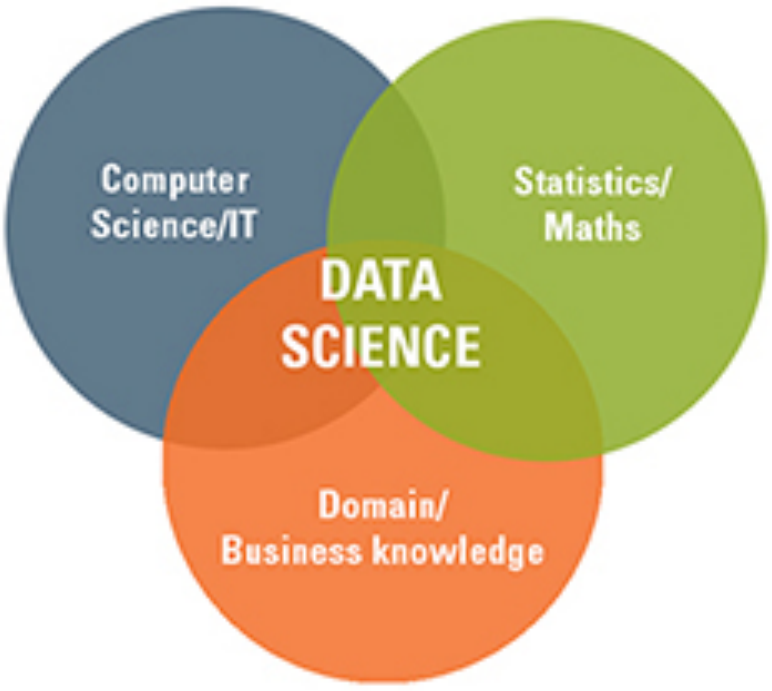

Figure 3.1: Data science subject Venn diagram

Figure 3.2 from Warwick University [52] confirms the cross-disciplinary nature of a Data Science program. These programs are heavily blended programs relying on skills from the math and computer science programs.

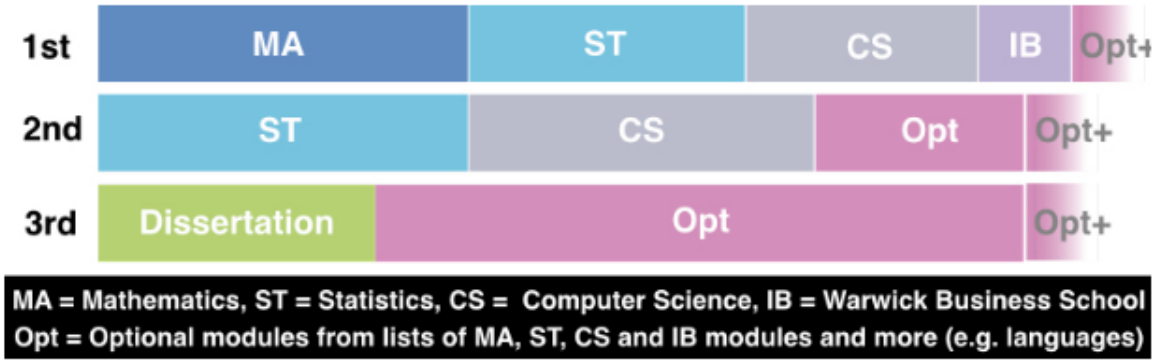

Figure 3.2: Data Science Course Structure

Conway [18] in Figure 3.3, confirms the need for domain expertise in data science. Conway states that domain experience combined with math and statistics is simply traditional research since this area does not allow for deep learning. This deep learning is made possible when domain experience and math and statistics are combined with computational power, data mining, and machine learning algorithms that are possible when leveraging computer science techniques. He called the combination of computer science and domain knowledge the "danger zone" because you can get an answer but there is no way to prove if it is statistically significant or if this is just a random observation which could cause a business to use this information for an invalid conclusion. The area that combines computer science and math/statistics Conway calls Machine Learning. We can improve the algorithms that analyze the data but we must be sure we are working on a meaningful problem. Only when we combine these three domains do we achieve data science. The combination of machine learning and computational processing, math and statistics, and domain knowledge allows us to utilize the mountains of data available and provides us the knowledge, tools, and algorithms to extract meaningful information that we can confidently use to provide solutions.

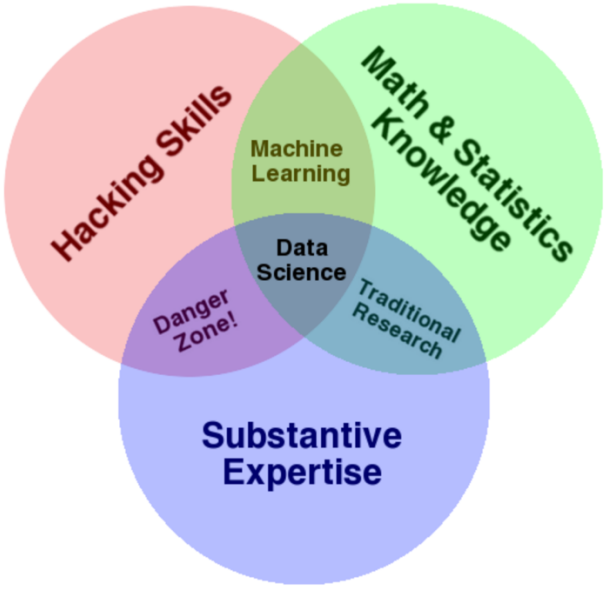

Figure 3.3: Data Science Venn Diagram, Conway

In "Big data and data science: what should we teach?" [57], Song et al began their discussion of a data science curriculum by first defining Big Data, data science and a data scientist. By examining their definitions, we can gain insight into the areas that need to be covered in a data science program. Big Data was defined by the 5 Vs: Volume, Velocity, Variety, Veracity and Value. Data science was defined as the study of the "generalizable extraction of knowledge from data." Data scientists focus on extracting actionable knowledge to solve problems, asking the questions that allow mapping to business goals and metrics, identify relevant data to solve this problem, select the technology and tools to process this data, perform analytics and visualize the results. The summary definition the authors presented was they

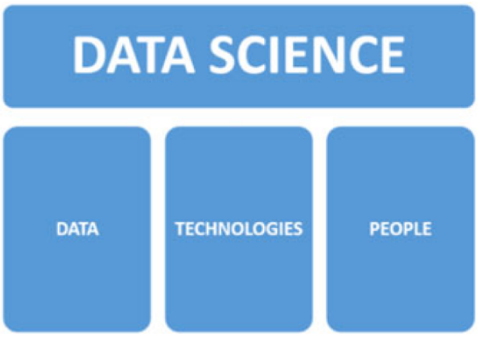

Figure 3.4: Data science and supporting pillars

assist in the automation of data-driven decision making. This is represented in Figure 3.4 showing the supporting pillars for data science: the data, technologies and the people. The authors then discussed the available programs for training data science. At that time, 2014, there were 9 programs offering data science programs. These programs were distributed across joint departments: Computer Science, dedicated Data Science and Business departments. Significant courses taught in these programs included Probability and Statistics, Data Mining, Machine Learning, and Data Visualization. The authors recommended training be based on the data analytics lifecycle model, Figure 3.5. This model is based on the 8 steps the authors felt were important in data science.

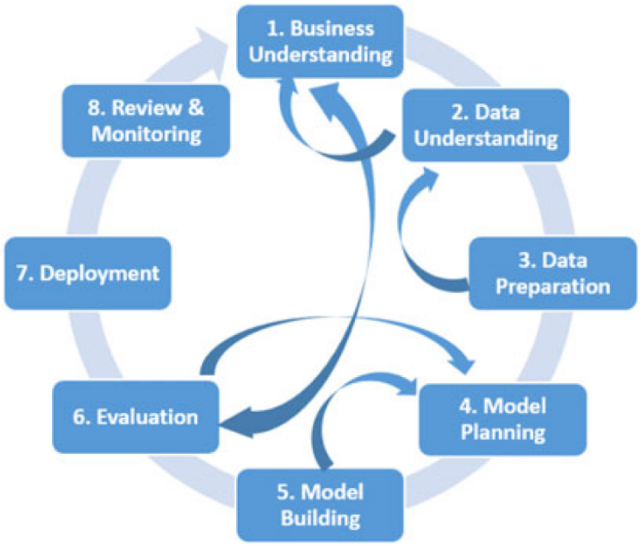

Figure 3.5: Data analytics lifecycle model

1. Understand the business problem.
2. Identify the right data sources and data sets required to solve the problem.
3. Prepare the data; extract, transform and load the data. Determine virtualization needed.
4. Plan the model for data, identify parameters, methods, and techniques that will be used in the model.
5. Build a model and techniques that will solve the problem.

6. Validate the model and results. Determine the success or failure against the business problem.
7. Implement the model in a production environment.
8. Review and monitor the performance of the deployed model.

Song et al recommended data science as multi-department; allowing each department to select the steps they are best suited to teach. They encouraged the use of programming languages such as Java and Python and statistical languages like R. Machine learning and visualization tools were deemed to be critical components. Other areas the authors felt were important included Hadoop and distributed parallel processing.

Ramamurthy [58] wrote in "A Practical and Sustainable Model for Learning and Teaching Data Science", data aggregation and mining have formed the core of most of the data advances. Ramamurthy's suggested program included four goals:

1. Understand data-intensive computing.
2. Study, design and develop solutions using data-intensive computing models such as MapReduce.
3. Perform predictive analysis and visualization using packages such as R and Google analytics.
4. Focus on methods for scalability using cloud computing infrastructures.

In 2017, Howe et al [59], published a paper "Data Science Education: We're Missing the Boat, Again" in which they discussed the current trend in data science education. Their suggestion was a four-year program in predictive analytics, machine learning and data mining. The course map they suggested included three courses in data science, six courses in computer science including data mining, artificial intelligence, and

data structures and algorithms, and eight math courses including calculus, linear algebra, statistics, and discrete structures. Large data set topics included creation, access, cleaning, analyzing, aggregating, organization, and visualization. Artificial intelligence topics included genetic algorithms, neural networks, Bayesian networks, intelligent agents, machine learning, and pattern matching.

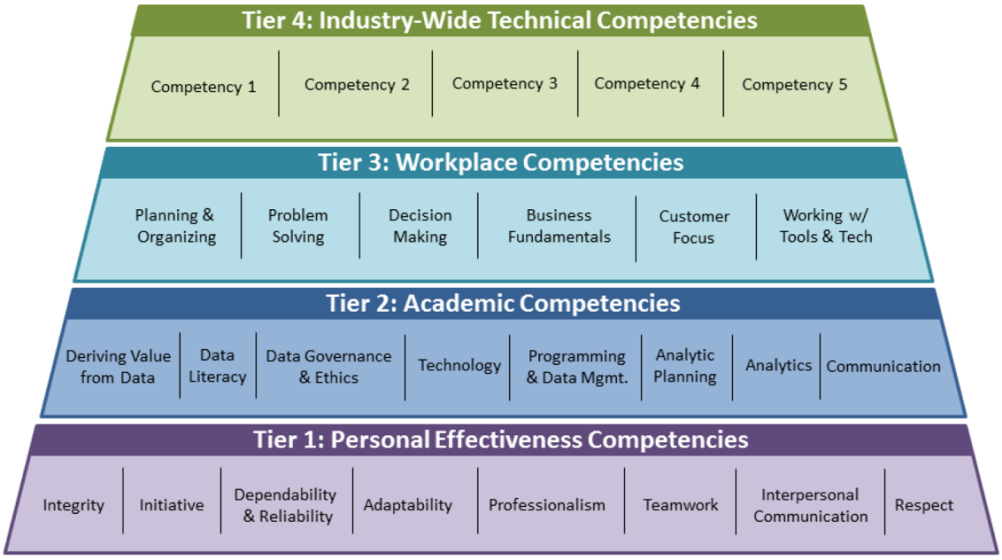

Figure 3.6: Competency map for data science graduates

The Business Higher Education Forum published a competency map for data science graduates [60]. In this report, they identified four tiers of skills required for a graduate in data science or data analytics. Figure 3.6 shows the four tiers and their associated skills. Based on the information contained in this report, we can identify critical skills required to be taught in a data science program. Tier 1 is composed of soft-skills that should be present in a graduate regardless of whether the program is a data science or any other program. Tier 2 contains foundational skills required by a data science graduate. Tier 3 skills focus on advanced problem-solving skills utilizing the

academic skills learned in Tier 2. Tier 4 leverages the lower tiers and combines these skills with domain specific knowledge to deliver meaningful analytics for the specific business problem. This report showed the tight fusion among the different areas of data science, computing, analytical and statistical skills, and domain knowledge. Focusing on the academic skills needed, we will focus on the skills covered in Tier 2 and the academic disciplines that must be used to teach these skills. These skills are:

1. Deriving value from data
    a. Students need to understand that data is a resource and the potential uses of this data to solve problems.
    b. A data mining course would allow students to understand data and its collection, cleaning, transformation, and application to a given problem.
2. Data literacy
    a. Identify and understand common data sources, structures and types of data.
    b. Again, a data mining course should teach students about the data lifecycle, structured, and unstructured data.
3. Data governance and ethics
    a. Students should be able to identify data governance issues, security, and privacy aspect of data.
    b. We feel that this should also be covered in a data mining course as well as in a Professional Practice and Ethics course.

4. Technology
    a. Students should be able to discuss different data manipulation technologies and their appropriate usage. They must be able to process, analyze, and present data to solve a given problem.
    b. The authors have broken this problem into two sub-problems. The first deals with the use of databases to store and process large data sets. The second applies data engineering principles to solve this problem. This deals with data that may be stored in a distributed system, an unstructured format, or any data that needs to go through ETL processing. This sub-problem should address other processing tools such as Hadoop, MongoDB, and emerging data science technologies. Students should be able to implement solutions using these tools.
    c. This area can be briefly covered in existing courses but will probably require the creation of new courses in data science and analytics focusing on the second sub-problem.
5. Programming and Data Management
    a. The authors defined this skill as the identification of "processes and mechanisms commonly used to retrieve, assess, enrich, manipulate and amalgamate data and applying them appropriately."
    b. We believe that this is part of the data science course mentioned above. This would be the practical application of the knowledge gained in the data science course.

6. Analytic planning and Analytics
    a. Examining and interpreting the requirements and data available to identify possible analytics-based solutions. Applying techniques to solve the problem.
    b. We believe these two skills should be combined to make a meaningful learning experience for the student. Simply planning the analytics will not give the student the foundational knowledge needed to master these two skills.
    c. This area points to a capstone course that would teach students how to apply all of the concepts that they have learned to solve data analytics problems. The student would need to identify the critical pieces of data, the information that is trying to be extracted, the technologies needed and the trade-offs each technology would need, the implementation of the solution and finally, the presentation of the analysis using the appropriate format.
7. Communication
    a. Create reports that will convey the results of the data analysis.
    b. This should be covered in a visualizations course where students learn appropriate ways to present the analysis. This could include tables, graphs, presentations and other means of communicating this information.

Tier 3 skills should be included with the above academic skills. This will require faculty to explain usage of these techniques in industry and help them to understand how these techniques can be used to solve business problems. While these are identified as critical skills, we believe that most students will only have a cursory knowledge of these skills and will learn them as they actually apply the Tier 2 skills.

Data Analytics has become a critical skill in all areas today. In his article, "Leading Educators: All Students Need Data Analytics Courses" [61], Abdul-Alim quoted Miami-Dade College president Eduardo Padron, "I really believe it applies everywhere" when Padron was discussing data analytics. This belief was also stated by the Business-Higher Education Forum in their report, "Investing in America's Data Science and Analysis Talent." [62]. In this report, the authors state while employers expect 69% of their employee candidates to possess data science and analytics skills, only 23% of college and university leaders state their students will have these skills.

According to ABET [30], an Information Technology program specializing in Data Scalability and Analytics should have the following competencies:

1. Appropriate data analysis methods.
2. Data preprocessing techniques.
3. Big data platforms – Hadoop, Spark and tools R, and RStudio. MapReduce and SAS to analyze data.
4. Data-intensive computations and streaming analytics on cluster and cloud infrastructures.
5. Impact of large-scale data analytics on organization performance.

Based on the proceeding information, a Data Science program could have drastically different outcomes depending on which school is hosting the program. We intend to address this from an Information Technology hosted program rather than focus on the Business or Statistics models. We intend to combine outcomes from other disciplines as appropriate to build a set of outcomes that cover all of the concepts we

believe should be incorporated into a Data Science program in Information Technology. Our proposed learning outcomes for a Data Science program include:

1. Apply data mining and computer science concepts to gather, clean, transform, and process data to build statistically based models to solve problems.
2. Assist organizations with the effective use of data, both structured and unstructured. Help organizations understand the impact of large data and the technologies used to process this data. Help organizations understand the myriad of data sources and data volumes that are available.
3. Select appropriate technologies, design, implement, and provide an analysis of data related problems.
4. Use predictive modeling to assist decision-making and to guide problem solution implementation.
5. Select and provide the appropriate reporting to assist in understanding analytics from a data-intensive analysis.
6. Apply interdisciplinary research and knowledge, and technology knowledge and skills to solve problems.
7. Ability to apply mathematical and statistical models and concepts to detect patterns in data, and to draw inferences and conclusions supported by data.
8. Ability to design, implement and analyze the software that manages the volume, heterogeneity and dynamic characteristics of large data sets and that leverages the computational power of multicore hardware and cloud computing environments.

According to ABET [63], a Computer Science program should have the following outcomes:

1. An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline.
2. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
3. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
4. An ability to function effectively on teams to accomplish a common goal.
5. An understanding of professional, ethical, legal, security and social issues and responsibilities.
6. An ability to communicate effectively with a range of audiences.
7. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
8. Recognition of the need for and an ability to engage in continuing professional development.
9. An ability to use current techniques, skills, and tools necessary for computing practice.

We believe there are many differences between a traditional Information Technology program and a Data Science program. We believe the biggest difference is the broader nature of a Data Science program. While an Information Technology program focuses on the use of technology to solve problems, the Data Science program

will also need to focus on the technology available, in many cases at a more technical level than a traditional IT program, and its use but will also need to focus on:

1. What data is available?
2. What are the methods to collect and munge this data?
3. What cleaning is required to make the data useful?
4. How is this cleaning accomplished?
5. What tools are available to analyze the data?
6. How can we leverage machine learning/automation to gain more insights from the data and simplify the analysis problem?
7. How do we handle the 5 Vs of data-intensive computing?
8. How do we present an analysis of a massive data set so that the analysis is actionable?
9. What domain knowledge do we need to utilize this data?
10. How do we validate the conclusions drawn from the analysis?
11. What data did we generate that can assist in solving this and other problems?
12. How do we ensure ethical use of the data both from a privacy and a security perspective?

The Data Science program must be more focused on data analytics and acquisition than on the computational theory and solution generation than an Information Technology program. The core technologies needed to perform many of these data science related tasks are not required in a traditional Information Technology program and therefore, based on our experience, we need to teach these foundational concepts to allow our students to be more effective in their data gathering, algorithmic design and

analysis. In addition, the Data Scientist needs to have knowledge of the domain in which they are working to enable them to make more meaningful observations and analysis, in other words, know what data is meaningful. While both of these programs do use technology, a Data Science program needs to go beyond solving computational problems and use technology as a tool to solve a domain problem. With the massive amounts of data being generated and available today, a Data Scientist must be able to leverage skills such as cloud computing, distributed and parallel processing, statistical analysis and significance, identification of meaningful outliers, machine learning to assist in the analysis, data identification and data cleansing.

We selected the outcomes from University of California, Berkeley as representative of the outcomes for a traditional statistics program. We selected Berkeley based on their top ranking as a statistics program, ranked number two and the top ranked program offering an undergraduate statistics degree. The objectives include: [64]

1. Fundamentals of probability theory
2. Statistical reasoning and inferential methods
3. Statistical computing
4. Statistical modeling and its limitations, and have skill in description interpretation and exploratory analysis of data by graphical and other means
5. Graduates are also expected to learn to communicate effectively

While a statistics program is designed to allow statisticians to look for significant differences in the data and identify areas of interest, the scope is simply too small when you look at the massive data volumes that are available today. Data sets comprised of millions of records and hundreds if not thousands of fields in each record are not

uncommon today. Traditional statistical analysis techniques will struggle to produce timely results from this data volume. With the large volumes, statistically significant results may be difficult to find. As stated in "Big Data and Management from the Editors" [28], "relying on p-values to establish the significance of a finding is unlikely to be effective because the immense volume of data means that almost everything is significant." This requires statisticians to deviate from a traditional statistical approach to finding significance and adopt a different paradigm. Data Science allows us to search for consilience – convergence of evidence from multiple data sets from independent sources to draw strong conclusions. These techniques are not adaptable in traditional statistical modeling and require us to expand to use new algorithms and modeling techniques for data analysis.

More importantly, most of today's data is unstructured which requires data preprocessing and manipulation to analyze. Because of the unstructured data, we must evaluate the data in its raw form and develop Extract, Transform and Load (ETL) processes that make the analysis possible.

With the large volumes of data, we need to be able to present conclusions in a meaningful manner which requires a different reporting mechanism where we can discover patterns in the data and also communicate these patterns to non-technical decision makers.

Finally, a Data Scientist must be trained to view the data produced by an analysis as a product itself which may lead to further analysis and increased value for decision makers. Simply completing an analysis is not the end for a Data Scientist but rather

should be a time of reflection to examine the data generated by the analysis for other useful insights.

The skills we believe must be present for a Data Science program graduate are as follows:

1. Ability to identify, gather, clean and analyze data to solve given problems
2. Ability to manage and program distributed systems using different analytical tools such as Hadoop, R, Pregel, Spark, Hive or other emerging technologies
3. Ability to interpret and present data analytics in a manner that allows non-technical personnel to interpret the results and apply this analysis in decision-making
4. Ability to apply domain knowledge to solve a problem whether this is through their own expertise or through utilization of subject matter experts
5. Ability to design algorithms using machine learning to assist in data mining and analysis
6. Ability to recognize information produced in analysis that is useful for future analysis and problem solving
7. Ability to combine mathematical, computational and domain knowledge to analyze data to solve a problem

CHAPTER 4

Infusing Data Science in an Information Technology Program

Before beginning to determine how to infuse data science in an Information Technology program, we must first establish the basic curriculum for an Information Technology program. We intend to first discuss the traditional Information Technology program, the ABET and ACM Information Technology curricula, and sample courses in an Information Technology degree. We will then discuss where and how data science can be injected into these courses to increase student knowledge of key concepts in data science including data mining, data analytics and data visualization.

The goal of an Information Technology program is to enable students to:

1. Explain and apply appropriate information technologies and employ appropriate methodologies to help an individual or organization achieve its goals and objectives.
2. Function as a user advocate.
3. Manage the information technology resources of an individual or organization.
4. Anticipate the changing direction of information technology and evaluate and communicate the likely utility of new technologies to an individual or organization.
5. Understand and, in some cases, contribute to the scientific, mathematical and theoretical foundations on which information technologies are built.
6. Live and work as a contributing, well-rounded member of society. [31]

An Information Technology program differs from a traditional Computer Science program in that it is Application, Deployment and Configuration oriented rather than Theory, Principles and Innovation oriented. This relationship is shown in Figure 4.1 [31]. This means many of the Information Technology courses are designed with business-oriented rather than research-oriented goals. Because of this difference, many of the traditional Computer Science courses are either electives or in some case not offered in an IT program. In the diagram in Figure 4.1, Information Technology is clearly shown to be more focused on application than theory. While the theory is important, the application of this theory is paramount. The majority of the students completing an Information Technology degree will proceed directly to industry where their ability to apply this theory will be a critical factor. While many of the Computer Science graduates also are destined for industry, Computer Science degrees are more focused on theory and principles rather than the application of these theories. Research in an Information Technology program is encouraged but not to the same level as in a Computer Science degree from a research-oriented institution. Because of these differences, many of the core Computer Science courses are either not taught or are taught as electives/sub-domain requirements which will add challenges to infusing data science in many of the essential courses.

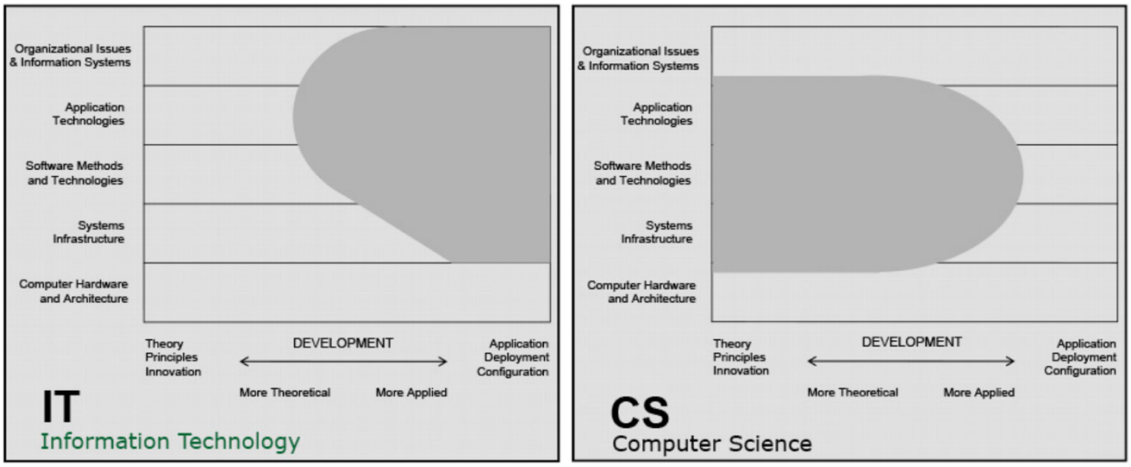

Figure 4.1: Comparison of Information Technology and Computer Science Theory vs Application

A traditional Information Technology program is built on a foundation of Information Technology fundamentals with five main supporting pillars that help the student become proficient in the two overarching principles of Information Assurance & Security and Professionalism. Figure 4.2 has been used to represent the Information Technology program since ACM published the first draft of the Information Technology Curricula in 2008. [30]

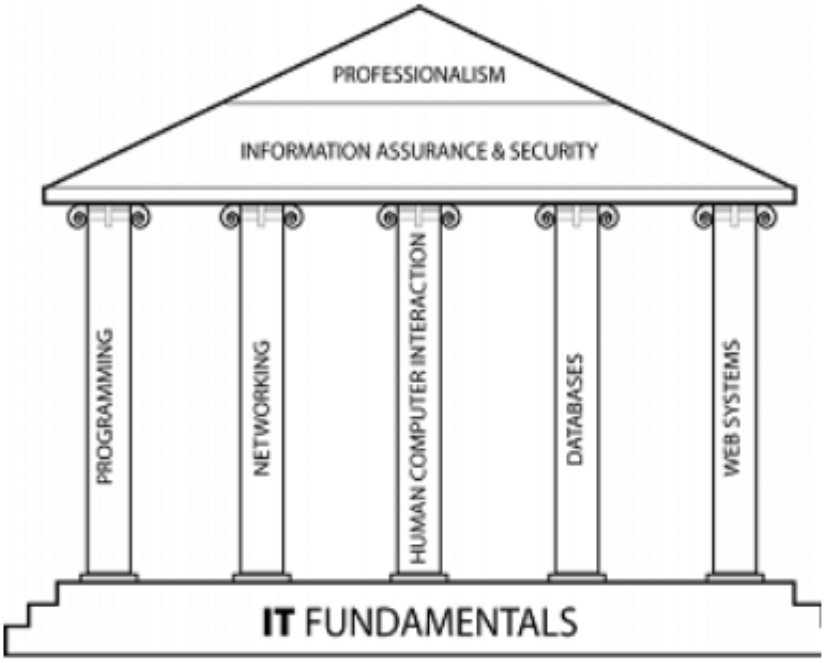

Figure 4.2: Information Technology Curricula Representation

Information Technology degree programs are intended to allow a student to specialize in the area they wish to pursue. This allows a student to select a domain and receive an Information Technology degree with a concentration in this sub-domain. At Georgia Gwinnett College, we offer five sub-domains: Software Development, Systems and Security, Digital Media, Enterprise Systems, and Data Science and Analytics. The ACM Information Technology Curricula recommends Information Technology Fundamentals comprise 15% of the course work in an IT degree of 120 hours. Essential areas include cybersecurity principles, information management, networking, system paradigms, user experience design, global professional practices, integrate system technology, platform technologies, software fundamentals and web and mobile systems. In addition to the recommended hours, a senior capstone course is required for every Information Technology concentration. [30] While not every student completes studies in

all of these areas, they must all achieve a minimum level of proficiency in these topics. All Information Technology programs are required to offer courses covering these fundamental concepts which include courses in programming, network database, systems analysis and design. These courses or their equivalent course offered under different names will be taught by every program that offers a compliant Information Technology degree. Within the Information Technology degree, the student is allowed to select a concentration to pursue that targets their career goals. The ACM Information Technology Curricula recommendation is for an additional 7.5% of a student's total hours to be spent in these courses, supplemental Information Technology domain, with a final 15% to be spent in Information Technology electives to help the student further prepare for their selected career path. [30] In our experience, the Information Technology supplemental hours are not sufficient to allow the student to master essential concepts. These courses will include advanced courses related to the student's selected concentration. For a software development student, these courses would include advanced programming and data structures, software development and design, testing and quality assurance, software development projects (the required capstone course). For a network or systems and security student, these supplemental courses would include information security, operating systems, advanced programming and data structures, and the information technology project (required capstone) courses. Other example concentrations include Cloud Computing, Web and Mobile Systems, User Experience Design and a new concentration in Data Scalability and Analytics. Each of these concentrations will have their own set of supplemental courses and offer electives to further assist students in mastering their selected areas. The Data Scalability and Analytics sub-domain provides

guidance for a data science concentration but does not apply this to any of the core fundamental areas of an Information Technology degree program.

Information Technology programs are focused on competency. As shown in Figure 4.3 [30], competency is made up of three key areas: Knowledge, Skills and Dispositions. Knowledge can be defined as being proficient in concepts and application of this knowledge to solve problems. Skills are the capabilities we develop through practice and learning interactions with others. Dispositions address our social interactions, work ethics and ability to complete tasks.

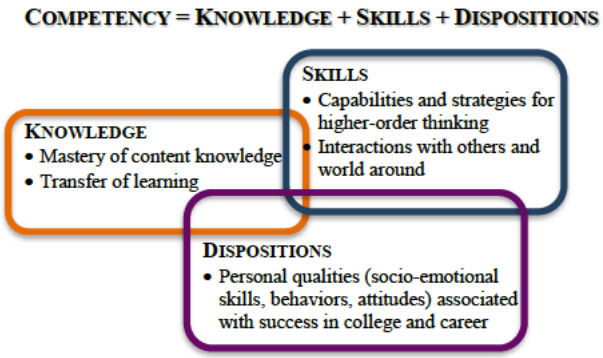

Figure 4.3: Competency mapping

These skills were identified by the Task Group on Information Technology as critical for a graduate to be successful following completion of an Information Technology degree. We must ensure that we keep this competency equation in our planning as we work to inject data science knowledge and skills into Information Technology courses.

Since there are many sub-domains, we need to look for opportunities to inject data science in the common core courses to expose all Information Technology students to these critical concepts. Some topics such as data visualization may have to be handled in

elective courses and briefly covered in the more advanced courses. Data intensive tool programming and use will need to be covered in some of the supplemental courses as well as in its own standalone course. The foundation needed for some of these concepts simply is not available in many of the essential courses and must be built on the concepts taught in these courses and in the more advanced supplemental courses.

The Data Scalability and Analytics sub-domain focuses on incorporating Data Science as the selected concentration. Key areas addressed are data collection, cleaning, manipulating, storing, analyzing and visualization. [30] Taking this as a starting point, we can use these key factors in the sub-domain to address the topics needed to be added to courses to provide data science skills and knowledge for all graduates from an IT program.

Specific areas covered in this domain are:

- Data science and its impact on industry. How to handle large volumes of data.
- Challenges of large-scale data, volume, variety, velocity and veracity.
- Data manipulation through the Extract, Transform and Load process. Data cleansing and knowledge extraction.
- Tools and techniques for data analysis including R, RStudio and Hadoop.
- Data governance – what security measures and access controls need to be in place? What are the ethics of sharing this data?
- Applications – What kinds of organizational problems can be solved using large-scale analytics? What is the data analytics lifecycle?

Interestingly, in the recently published, "Information Technology Curricula 2017" report, the list of driving forces from Information Technology innovation included mobile

applications, social platforms, user experience Internet of Things, cybersecurity and automation. [30] Missing from this list, in our opinion, was the impact data science and data intensive computing is having on the needs of our Information Technology students. While Internet of Things and social platforms data can be viewed as contributing to these needs, no clear driving force was associated with data visualization, data mining and data intensive computing.

ABET lists the following outcomes for an Information Technology graduate in its 2017-2018 proposal.

1. An ability to analyze a problem, and to identify and define the computing requirements appropriate to its solution.
2. An ability to design, implement, and evaluate a computer-based solution to meet a given set of computing requirements in the context of the discipline.
3. An ability to communicate effectively with a range of audiences about technical information.
4. An ability to make informed judgments in computing practice based on legal and ethical principles.
5. An ability to function effectively on teams to establish goals, plan tasks, meet deadlines, manage risk, and produce deliverables.
6. An ability to identify and analyze user needs and to take them into account in the selection, integration, evaluation, and administration of computer-based systems. [IT] [65]

ABET does not specifically list required courses or domains but rather lists expected student outcomes. These outcomes do not leverage any particular technology

but rather the student's ability to perform using whatever technologies they choose. Therefore, when we begin discussing the infusion of data science into existing courses, we need to ensure each of these outcomes is met by students who receive the data science training.

The outcomes for ABET as related to data science education can be re-stated as follows:

1. An ability to analyze a data-intensive problem; identify the correct tools, techniques, data cleaning and data transitioning required to solve the problem
2. An ability to select the correct tools and processing to implement a solution to the data-intensive problem and correctly evaluate the implemented solution to determine if the correct solution was implemented
3. Communicate effectively using data visualization with technical and non-technical audiences
4. Select the correct data governance models resulting in ethical handling of all data associated with the data-intensive problem
5. Work effectively with their team providing guidance and implementation to manage risks and produce the deliverable necessary to complete the data-intensive problem
6. Identify information and user needs providing guidance on data selection, data transitioning and implementation of large-scale data related problems enabling users to select the best solution to solve their problems

The general Information Technology program should meet the following requirements as stated in the ACM Information Technology Curricula from 2008.

- Knowledge of the scientific method – The Information Technology graduate must gain experience with the scientific method through "direct hands-on experience with hypothesis formulation, experimental design, hypothesis testing, and data analysis"
- Familiarity with application domains – Information Technology students cannot be technologist only; they must be able "to work effectively with people from other disciplines"
- Communication skills – Information Technology graduates "must be able to communicate effectively with colleagues and clients"; both technical and non-technical colleagues and clients
- Teamwork skills – Information Technology graduates must be able to work effectively in teams
- Knowledge and skills of becoming a contributing member of society – Information Technology graduates must have the appropriate skills to function effectively and cordially in society
- Pervasive themes – Information Technology graduates must "acquire a skill set that enables them to successfully perform integrative tasks, including user advocacy skills, the ability to address information assurance and security concerns, the ability to manage complexity through abstraction, extensive capabilities for problem solving across a range of integrated information and

communication technologies, adaptability, outstanding interpersonal skills, high ethical standards, and professional responsibility" [66]

These requirements have helped focus Information Technology more in the application solution domain than in the theoretical and research area. While many of these skills are useful in research and theory-oriented programs, these become paramount in the application solution domain. Based on the requirements listed above, IT students pursuing data science concentrations should spend time studying other subjects in the field in which they intend to pursue data analytics positions. We believe this will eventually require students in a Data Science concentration to take a subset of their courses in areas outside the Information Technology framework, for example, in biology, mathematics, accounting and finance and other subjects. This has been successfully implemented at the College of Charleston where students are required to select a cognate with between 15 and 22 hours in that subject area. [67]

Because an Information Technology graduate needs to be able to communicate in their chosen domain, as we bring knowledge of data science into existing courses, we will need to coordinate with faculty across campus to allow them to introduce concepts that are crucial to data science in these courses. This integration would best be done in the General Education courses which allow us to broaden the exposure for all students to data science and its application.

Some of the core courses for an Information Technology curriculum based on the ACM model would include courses covering the following topics:

- Information Technology Cornerstone – a course for developing basic computer skills, problem solving, study habits and time management. This course would

also cover basic concepts in hardware, software, networking, internet usage and security

- Introduction to Computer Systems – basic programming and how a computer works. Binary arithmetic and boolean logic
- Computer programming – introduction to object-oriented program design and development
- Fundamentals of Web-based Information Technology – web technologies including distributed architecture, database concepts, networking and server development
- Database Principles and Applications – database theory, data modeling and designing application databases. Query language and data security are also covered
- Operating Systems – configuration, file systems, security, administration and networking
- Computer Networks – local and wide-area networking. Workgroups, routers, hubs switches, and network administration
- Information Assurance and Security – computer security principles. Incident prevention and management
- Senior project/capstone – project management, teamwork principles, intellectual property and professional technical literature review [30]

These courses or their equivalent appear to be ideal places where we can infuse many of the data science technologies to increase student awareness of data science theory and technologies.

Based on these re-stated outcomes and the ACM Information Technology curricula, we can now begin looking for opportunities to infuse data science into existing Information Technology courses.

The following section includes a listing of the core courses identified above and the Data Science concepts we feel are appropriate for introduction in each course. We are basing these recommendations largely on the Data Scalability and Analytics sub-domain as well as our experience in Data-Intensive and Distributed Systems courses. In addition, we are leveraging more than 10 years of experience as Information Technology faculty members and the knowledge gained from teaching many of these courses multiple times. We also evaluated the projected outcomes of each course against the ABET objectives we proposed for Data Science above.

- Information Technology Cornerstone
    - Perspectives and impacts
        - What is data science?
        - Why has data science become so important to businesses?
        - What is the history of data science?
        - What are the tools and applications used in data science?
        - Where does all of this data come from?
        - Define volume, variety, velocity and veracity. Many industries are now adding value as a fifth V. [68] [69] [70]
    - Data management
        - An introduction to data mining.
        - Purpose of data mining.

        - Introduction to techniques.
    - Evaluation
        - This would be very introductory material. Outcome expectations would be for students to be able to understand core concepts relating to data science and articulate these. In depth knowledge would not be available following this course; simply a general knowledge of large-scale data problems, analytics and basic principles in data mining.
        - Students would be expected to understand the technological journey that has led us into data-intensive computation and the variety of sources producing data in today's technology infrastructure. This would include a working knowledge of the 5 Vs and why these are important in data science.
- Introduction to Computer Systems
    - Data Management
        - Data warehousing.
        - Structured and unstructured data.
    - Data governance
        - Ethics and legal issues surrounding storage and manipulation of data.
        - How does a large-scale data application change how data must be handled?
        - What are the data sources and how should these be safe-guarded and made available for use?

- Data cleansing can be introduced here to remove unneeded identifying information.
- What kinds of security can be established to allow organizations to make use of their data while preventing disclosure of either sensitive information or proprietary information that could harm the organization or its customers?

- Large-scale data challenges
  - Volume, Velocity, Variety, Veracity and Value.
  - Data sources – sensors, social media, retail transactions, etc.
  - Platforms available for data analytics. Advantages and disadvantages of each analytic platform.
- Methods, techniques and tools
  - Students should be encouraged to research existing and new tools and technologies. A possible assignment would be to have the students divide into groups and present emerging data science technologies.
- Applications
  - Commercial applications and the reasons for data analytics should be discussed in this course.
  - What impacts can large-scale data analytics have on a company's decision-making process?
  - Define an organizational problem that could be resolved using large-scale analytics. What would be the data sources?

- Evaluation
  - The student should be able to recognize situations where large-scale analytics would be useful and determine possible data sources that are applicable to this problem. Data selected should not be limited to structured data alone but also utilize unstructured data.
  - The student should be able to discuss different tools, techniques and platforms used in data analytics.
  - The student should understand ethical and legal use of data and the necessity for protecting this data.
  - The student should be able to discuss the 5 Vs and different data sources that would be useful in solving the observed business problem.

- Computer Programming
  - Data science fundamentals can be introduced in early level courses and concepts can be introduced. We do not believe we can add data science specific programming, application development or tools to these early courses. The material taught in these courses is difficult for students to master and adding more would result in a higher rate of failure for these courses. We can however, discuss how data science is changing the Information Technology landscape and some of the challenges associated with it. We can discuss tools and re-enforce concepts covered in earlier courses. In the more advanced courses, we could introduce distributed systems concepts and programming. This could include calling API's on different systems, using

sockets to communicate with remote devices, and accessing databases from student applications.

- Methods, techniques and tools
  - Introduce Processing in the earliest course to help students with visualization concepts. Processing has been used to provide graphical feedback at an early stage in programming courses but this could lead to discussion about its use in data analytics and visualization.
  - Introduce R in the upper level courses to prepare students for statistical analysis using R in follow on courses.
- Applications
  - In the advanced course, discuss large-scale data analytics problems and techniques to support solving problems in this domain. This could include distributed programming, data structures that provide support to solve large-scale problems, and proper selection of tools and languages.
- Evaluation
  - For introductory courses, students should be able to discuss data science, why it is important and tools that can be used.
  - For introductory courses, students should be able to develop rudimentary programs in both an object-oriented language and in a visualization-based language like Processing.
  - For advanced courses, students should be able to programmatically call data sources including both relational and NoSQL data sources.

        - For advanced courses, students should be able to utilize data analytics tools to solve simple large-scale data problems.
- Fundamentals of Web-based Information Technology
    - Large-scale data challenges
        - Recognize the variety of sources available for data collection such as social media, news feeds, eBusiness transactions, click counts, etc.
    - Methods, techniques and tools
        - What are the impacts of cloud computing? Widely distributed data and processing?
        - Design and develop solutions to harvest data from various sources including those listed above.
        - Develop a basic understanding of different tools such as Hadoop, Spark, Hive, Crunch and SAS and possibly R.
    - Data governance
        - Identify issues surrounding appropriate use of data; the legal and ethical consequences of using such data.
        - Identify security and privacy issues surrounding cloud computing and web available distributed data.
    - Applications
        - Develop rudimentary applications to store, retrieve and manipulate large-scale data in a web or cloud environment.
        - Examine the differences between cloud and web implementation and local cluster implementation.

- Evaluation
    - Students should be able to explain data source selection, location and processing associated with a large-scale data analytics problem.
    - Students should be able to explain the ethical and legal use and ramifications of data used in a large-scale data analytics problem.
    - Students should be able to discuss rudimentary security issues and tools used to safe-guard data.
    - Students should have a basic understanding of various large-scale data tools, their advantages and disadvantages.

- Database principles and applications
    - Traditional database courses do not address unstructured data or the use of NoSQL databases. With this in mind, we should modify this course in the following manner.
        - Data management
            - Increase exposure to NoSQL databases.
            - Develop extract, load and transform scenarios for unstructured data.
            - Build basic data cleansing applications allowing students to understand the process of data substitution, data removal, redundancy removal and data correctness and completeness validation.
        - Data governance

- Provide training for students to assist them in recognizing sensitive data.
- Provide training to allow students to understand data protection measures.

- Applications
  - Provide training and experience in utilizing data analytics to solve an organizational problem.

- Evaluation
  - Students should be able to present a set of data storage alternatives for an organizational large-scale data analytics problem, discuss the advantages and disadvantages of each, select the best alternative and implement this data storage solution.
  - Select an appropriate data governance model allowing for secure and ethical storage of large-scale data.
  - Students should be able to recommend and implement data extraction, loading and transformation to support a given large-scale data problem.

- Operating Systems and Computer Networks
  - Applications
    - Introduce distributed systems concepts.
    - Provide training on cloud and distributed computing.
    - Provide training on developing cloud/distributed applications and the challenges associated with this type of application.
  - Large-scale data challenges

        - Train students in advantages of utilizing distributed systems to allow for parallel processing of large-scale data sets. This should include a basic introduction to some of the tools used for large-scale processing such as Hadoop, Spark, Hive and/or R.
    - Evaluation
        - Students should be able to discuss basic distributed systems concepts.
        - Students should be able to discuss the benefits of utilizing cloud and distributed applications.
        - Students should be able to discuss the challenges of using distributed applications and strategies to overcome these challenges.
        - Students should be able to discuss and implement a simple program using data analytics tools.
- Senior Project/Capstone
    - This course should be the culmination of the Information Technology program. In this course, the student should demonstrate their understanding of the material they have studied in preceding courses. Keeping this in mind, this course should expand the student's ability to utilize data analytics tools, techniques and competencies.
    - Perspectives and impact
        - Present real-world data analytics problems. Assist students in designing data analytics solutions. Assist students in selecting appropriate data sources to solve this problem.

- Large-scale data challenges
  - Assist students in selection of the appropriate platform and tools to process this data.
  - Assist students in examining multiple data sources related to a problem including social media, sensor data, financial records, retail/eBusiness records, or any other relevant data source.
  - Assist students in comparing the different platforms, data sources and techniques to solve the selected problem.
- Data Management
  - Assist students in the Extract, Transform and Load process to generate meaningful data to solve the selected problem.
  - Assist students in utilization of data mining techniques for the selected problem.
- Methods, techniques and tools
  - Teach students new tools and technologies for data analytics.
  - Assist students in their selection of the tool and technology to solve a particular problem.
  - Encourage student to research different tools and technologies helping them gain insights to emerging technologies.
- Data Governance
  - Assist students in selecting proper data security and ethical practices for data.

        - Discuss ethical, legal, and social issues surrounding data and data collection techniques.
    - Applications
        - Develop large-scale analytics programs to solve the business needs of the selected problem.
        - Apply the data analytics lifecycle to ensure students understand data collection, cleansing, storage and finally data destruction. Assist students with security and governance policies during this lifecycle.
        - Assist students in visualization of data analytics results and the communication of these results to a "business customer".
    - Evaluation
        - Students should be able to build, manage and communicate the results of a large-scale data analytics program.
        - Students should be able to discuss security and privacy issues surrounding the collection, storage, processing and deletion of data.
        - Students should be able to discuss different data analytics platforms, the advantages and disadvantages of each and assist in the selection of a data analytics tool to solve a given problem.

There are several remaining areas that we do not believe fit well into an existing Information Technology curriculum. These include data visualization, machine learning and data mining. While these topics can be given cursory treatment in the existing courses, there is simply too much material that needs to be taught in each of these areas to

fit into the Information Technology curriculum. They will have to be taught in specialized electives for those students interested in pursuing data science.

Many Information Technology programs include an Artificial Intelligence course which could be enhanced to cover the material required for data science. This course should include coverage of inductive learning, decision trees, neural networks, and Bayesian learning techniques. The following are outcomes we would expect from such a course:

- Inductive learning should discuss training data and the theory expecting unseen data to follow the behavior of a sufficient training set.
- Decision trees
    - Students should understand how the various attributes can lead to a tree structure capable of providing a learned function that solves the discrete function contained in the training data.
    - Students should be exposed to the greedy algorithm for selecting the best available choice and working through the process to solve for training examples based on the remaining best attributes.
    - Students should be able to identify problems for which a decision tree would be appropriate.
    - Students should be able to discuss and utilize post-pruning techniques.
- Artificial Neural Networks
    - Students should recognize which types of problems are best suited for neural networks.

- Students should be able to explain and use perceptrons and how to adjust weights to improve learning rate.
- Backpropagation should be discussed time permitting.

- Bayesian learning techniques
  - Bayes Theorem.
  - Naïve Bayes Classifier.
    - Students should be able to use the Naïve Bayes classifier to utilize learning steps to generate conditionally independent probabilities for use in calculating the posterior probability.
    - Students should understand the restrictions places on Naïve Bayes.
  - Bayesian networks.
    - Students should understand the difference in a Bayesian network and a Markov random field. They should be able to discuss the differences and why the directed nature of a Bayesian network, or belief network, is important.
    - Students should be able to traverse a Bayesian network and calculate the posterior probability.

A data mining course, based loosely on Introduction to Data Mining [71], should be added. This would include the history of data mining, types of data that might be used in today's environment, the data mining process, machine learning concepts, classification techniques, association analysis, and anomaly detection. The following information would be a minimum coverage of these topics:

- History and basic overview.

- What is data mining?
- Why is data mining important?
- Knowledge Discovery in Databases history.
- Challenges.
  - Data volume.
  - Enormous complexity in data sets, many rows and many columns.
  - Unstructured data.
    - Heterogeneity of data.
    - Variety of data sources.
  - New analysis techniques required.
    - Limitations of existing analysis techniques.
    - New thought process required for hypothesis generation and examination.
- Origins of Data Mining.
- Data Mining Tasks.
  - Predictive modeling.
    - Purpose and scope of a model to handle classification and regression analysis.
  - Cluster analysis.
    - Grouping of data to allow more efficient analysis of similar data elements.
    - K-means and nearest neighbor.
  - Anomaly detection.

        - Identification of outliers and how these can lead to significant findings.
- Data.
    - Types.
        - Qualitative vs quantitative.
        - Time series.
        - Attributes of data and measurement scales.
    - Quality.
        - Error identification.
        - Cleaning.
    - Transformations.
    - Analysis.
- Classification.
    - Techniques.
    - Issues.
        - Overfitting.
        - Underfitting.
    - Rule-based classifiers.
- Association analysis.
    - Binary representations.
    - Frequent item sets.
    - FP-trees.
    - Evaluation techniques.

- Cluster analysis.
    - Meaningfulness.
    - Summarization.
- Anomaly detection.
    - Statistical approaches.
        - Parametric models.
            - Univariate Gaussian distributions.
            - Multivariate Gaussian distributions.
        - Model-based.
            - Models built for normal classes.
            - Models built for normal and anomalous classes.
    - Proximity based approaches.
        - Distance-based score based on the distance to it nearest k neighbors.
        - Density-based score to identify anomalies by sparseness.

Data visualization would be the last course we would add to this program. Rather than focusing on a specific tool, we believe this course should be focused on why we need to use visualization, what are the key cues when viewing a visualized data set, what are the formats, and the advantages/disadvantages of each format. Additionally, we should discuss the ethics in presentation and how to avoid skewing the presentation of data. Specific topics to be covered include:

- Visualization rationale.
    - More easily communicate information.
    - Allow for pattern recognition for analysis.

- Visualization cues.
    - Title – Does it explain the purpose/data being displayed?
    - Position – Where are values in relation to other values?
    - Visual cues.
        - Shape.
        - Encoding.
        - Colors.
        - Size.
        - Etc.
    - Coordinate system and scale.
        - Are they honest?
        - Do they allow the user to do effective comparisons?
- Basic visualizations that should be covered include:
    - Time-series.
    - Top-down.
    - Radial.
    - Maps.
    - Matrices.
    - Pie.
- New layouts are emerging and should be covered based on the latest technologies such as zoomable, clickable, and online visualizations. As new technologies emerge, these will need to be added to this course.

CHAPTER 5

Industry Perspectives on Data Science

5.1 Industry Literature Review

To formulate curriculum to prepare students for data science related tasks, we must first examine the roles and tasks these recent graduates must perform. To determine this, we have used a combination of industry papers, job posting, and interviews with various industry experts on their expectation for a newly recruited data analyst/scientist. It is important to note that there is not common agreement across all industries on what skills are deemed critical. Some companies are leveraging traditional statistics coupled with visualizations and tools such as Microsoft Excel. Others expect their analyst/scientist to design algorithms, code complex data intensive solutions, collect and clean data, and interpret the analysis and communicate their findings using both visualization tools and written/oral communication to convey these business insights.

In "Analytics: The real-world use of big data – How innovative enterprises extract value from uncertain data" [72], Schroeck et al state "the most effective big data solutions identify business requirements first and then tailor the infrastructure, data sources and analytics to support the business opportunity." One of the major concerns they described in this article was the lack of advanced analytic capabilities which they felt were required but were often lacking in an organization. Schroeck et al state that the impact of big data can only be maximized when companies have integrated the information foundation in their analytics efforts. They state that 75% of the respondents

in their survey were analyzing log data, sensor data, and other generated records to understand the performance of these systems. This requires understanding of basic technologies used for querying and reporting, data mining, and visualization. The biggest shortfall addressed in this paper is the critical shortage of analytical skills upon which big data effectiveness hinges.

451 Advisors in "The Cloud-Based Approach to Achieving Business Value from Big Data" [73] state Data Scientists "need to be able to use tools such as Python and R to analyze data but can also apply these skills to create prototypes of new products and services." Other areas identified by 451 Advisors included business analytics skills, ability to interact with external data sources and integrate them with internal systems, data governance, security, and responding to change. This article proposed moving many analytic tasks to the cloud instead of hosting internally bringing even more focus on the technology skills required for cloud and distributed computing.

IBM in "Big data and analytics in travel and transportation" [74], discussed the impacts of big data processing on the transportation industry and again emphasized advanced "analytic technologies and techniques as enablers for this industry to extract insights from data reaching previously unachievable levels of sophistication and accuracy." Tasks discussed in this industry included providing insights from both current and historical data, predicting and detecting competitor's pricing and allowing the company to respond to these changes, using sensor data as well as data from various sources to perform predictive analytics.

In "Big Data in manufacturing: A compass for growth" [75], Infor.com predicted big data will "transform outdated manufacturing facilities into highly automated, efficient

powerhouses." This is the result of data being collected from traditional sources such as customer surveys, being combined with smart sensors, the Internet of Things and external data such as user groups and social media. This influx of data will allow trends to be predicted and needs to be anticipated. This should result in predictive analysis to help form the blueprint for future actions. The data analysts must be able to interface with these different systems and utilize data intensive tools to assist in collection, munging, and presentation of this data.

Carnelley et al in "Big Data: Turning Promise Into Reality" [76], identified different benefits for three industries: healthcare, telecommunications, and financial services. They also identified the broad range of skills necessary for these tasks. The skills identified are more business oriented than those we have seen in other papers. These skills include:

- Multidimensional analysis
- Visual discovery
- Advanced analytics
- Data collection, integration and preparation
- BI, app and dashboard development
- Performance measurement
- Governance
- Reporting skills

Even with a less technical approach, core skills were still needed to handle the analytics, data collection and preparation, and business intelligence development.

Guiterrez et al in "Big Data Industry Perspective for 2017" [77], discuss the application of data in today's industry. They state "the IT infrastructure and skills are at the core of the transformation to analytics at scale." Data provides insights into customers and as more data becomes available, businesses are better able to serve their needs. With the proper usage of data, businesses gain many more touchpoints with their customers, allowing them to improve the customer experience. The data analysts provide the ability to maintain this information flow and it must be done on a real-time basis. Chuck Pieper stated, "There is a real gap between their data visions and the ability of their enterprise to move data horizontally throughout the organization. In the past, big data analysis has lagged in implementation compared to other parts of the business being transformed by advanced technology." Streaming analytics are becoming more important and will be the default for enterprises allowing the business to make more informed decisions. For analysts, text analytics will be replaced with Machine Learning allowing automation of many data analytics tasks.

In 2018, Steunwegg-Woords published an article "What Skills Does a Data Scientist Actually Need? A Guide to the Most Popular Data Jobs" [78], detailing critical skills required for what at that time was Glassdoor's top job for the precious three years; a Data Scientist. Skills required for a Data Analyst were:

- The ability to answer ad-hoc questions from the business team
- Ability to prepare visualization in the form of a dashboard or other appropriate communication mediums

- Tools required
    - SQL
    - Tableau
    - Excel
    - SAS

For a Data Engineer, this changes slightly. The data engineer's primary responsibility is to make data accessible to the data scientist and analyst. Tools identified were Python, Java, and Scala.

In 2019, IT Career Finder published their definition of a Data Scientist [2]. They state "Data Scientist analyze business data for actionable intelligence." The day to day activities included:

- Perform data mining – model and hypothesis generation must be done to support business goals
- Stay current with emerging technologies
- Have strong business, technical, mathematical, and statistical abilities
- Develop customized algorithms
- Possess strong oral and written communication skills
- Leverage tools such as Hadoop, SQL, Python, Unix, PHP, R, and Java

Educational requirements for a Data Scientist vary from company to company. Many require an advanced degree while others will accept candidates with undergraduate degrees in computer science or a related field. [2]

5.2 Interviews with Industry Personnel on Data Science

Mr. Rob Houser as the Senior Director of Product Management at Sage Software, Inc. stated he had "an entire team that focused on building tools that leveraged the various data sources we could access." One of his more prominent data analysts had an undergraduate degree in computer science which enabled him to gather data from various sources, normalize the data and create reports based on this data. This required him to investigate and learn new processing tools to manipulate and collect this data. Since one of the major roles was to interpret the data and ensure meaningful information was presented, this analyst relied heavily on his MBA to help provide business insights. Later when Mr. Houser transitioned to the Senior Director of User Experience at CareerBuilder, he had slightly different expectations for his data analysts. The analysts at CareerBuilder were focused on the extraction of data from unstructured sources, candidate resumes, job postings from different companies, and press releases both from CareerBuilder and other sources. By utilizing data collection and data intensive tools, CareerBuilder was able to search for emerging job trends and collate this unstructured data and convert it into meaningful business analytics. This required analysts to "understand real-world problems that recruiters were trying to solve and then find ways to leverage the data that CareerBuilder had." These analysts wrote queries against traditional relational databases but also had to mine unstructured data requiring technical knowledge in data intensive tools and machine learning. Currently, Mr. Houser is the Senior Director of Product Design at Finsync where he is again searching for data analyst who can map knowledge from data collected using "big data tools connected to various sources of data: SQL, statistics, and data visualizations." Many of these sources require

the analyst to be able to programmatically extract information. “Analysts had to be highly analytical, creative, and data oriented.”

The following is an opinion paper on Data Science from Mr. Michael Weaver, Executive Director Global Consulting Operations, Teradata.

“Data Science is finally starting to come of age, moving from its childhood into a likely confused adolescence. Formerly Data Science has been a lightly defined, largely misunderstood marketing catchphrase. With the advent of solid university level curricula Data Science is finally coming into its own, moving from art to a more practical science. The Data Scientist role is yet in the process of solidifying its practical role integration. Our belief that the Data Scientist role should purposely overlap with the key roles of Business Analysist and Systems Architect. Also, the Data Scientist role must understand and be conversant with multiple adjacent and related roles, as seen in the figure below. Data Science is a multi-disciplinary function but must resist the temptation of trying to be all things to all people which will only confuse, or sub-optimize its role and function.

At its core, the role of the Data Scientist is to create or extract value from data in the simplest manner reasonable. To do this the Data Scientist must bridge two worlds. First, they must work together with the Business Analysis and the end user customer to understand the customer’s business problem, needs and objective(s). A solid understanding of general business, business process and problem decomposition is critical to accomplish this. Second, understanding data, which in turn has two major components: first to understand what data is available in what form or structure and second, what data is needed but is not readily available so must be sought or derived. Bridging these two worlds together is the core function of the Data Scientist.

The challenge with creating a Data Scientist is that the Data Scientist needs to understand enough of both Business and Data worlds to build the desired bridge, that of extracted data value. It would be easy to say that the Data Scientist needs to be an expert in all roles, but that is not practical. That said, the Data Scientist should have a reasonable working understanding of key data structures, i.e. 3rd normal form, snowflake, etc. and key data tools, i.e. Hadoop, Spark, Machine Learning, etc. so that they can create the requisite algorithms and analytic models to extract the needed data value. Developing a curriculum at an interrelated component level and accomplishing this set of objectives has been and is the focus of universities and their academic staff today, with strong practical contributions of industry practitioners. This has led to the additional functional componentization of the Data Science subject area to a set of related sub roles: Data Analyst, Data Architect, Data Statistician, Machine Learning Engineer, etc. which will need additional applied role definition."

As shown in Figure 5.1, Mr. Weaver believes the role of the data scientist combines many aspects of fields thought traditionally to be in the computer science realm but also expands to include those found in a business analyst. The role of the data scientist as proposed by Mr. Weaver includes the following key skills:

- Technical knowledge
    - Rudimentary IT skills enabling data collection
    - Programming
    - Data Structures
    - Database theory
    - Hadoop

- Spark
- Machine learning

- Communication skills
  - Visualization
  - Oral and written communication
- Business skills
  - Domain knowledge
  - Business acumen
- Statistical processing
  - Ability to recognize significant findings.

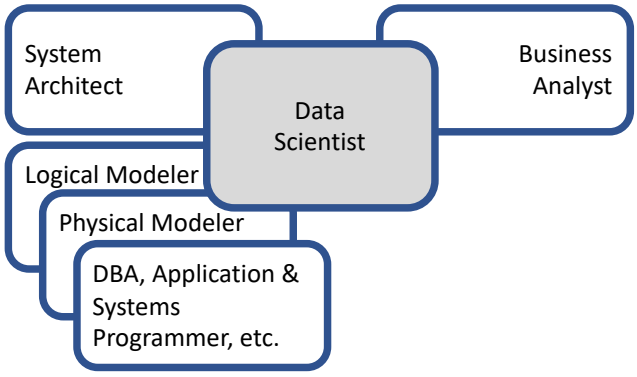

Figure 5.1: Data Scientist Roles per Mr. Michael Weaver

Mr. Brandon Huff, Director of Information Technology, Mr. James Smith, Data Strategist and Trainer, and Mrs. Laura Bazemore, Church Management Systems Administrator met and discussed the skills and duties of a data analyst/scientist. They identified key skills for a data analyst for their environment. These skills included:

- Tableau
- Power BI

- On the job experience – Internship
- Microsoft SQL Knowledge
- Church Management Systems
- Organizational understanding
- Communication skills/relational/trustworthy/team player
- Data is foundational to all systems
- Create data accessibility

Interestingly, although this is a completely different domain, many of the core skills identified by corporations were also present in the church environment. These common skills include domain knowledge, communication, creating and accessing data, technical skills primarily focused in SQL, and familiarity with the tools used for data presentation (Tableau and Power BI). Mr. Huff went on to state, "It's important for a data scientist to know their platform so they can become more of a team player. The analyst must have soft skills such as communication, being relational, and being a team player to be able to provide the product the company needs. A data scientist must think broadly allowing them to adjust to provide what is needed." Mr. Smith stated "A data scientist brings data to the foundation of everyday things. While many people want to generate cool reports, the critical factor is providing executives with the information they need to make decisions."

Mr. Doug Mulkey, Chief Technology Officer for Finsync had this to say about data science. "Any data analyst will need the skills necessary to consume data from various sources, manipulate data in different formats, and store data in a format that makes it easy to analyze and visualize. One of the main skills needed will be an

understanding of database technologies and SQL. They should have a good understanding of how to do bulk ETL, a basic understanding of what Star and snowflake schemas are, and the pros and cons of using a NoSQL database. Since not all data sources will be a SQL database some level of programming may be needed to consume and manipulate data coming from non-DB sources such as an API. Depending on the role, an entry level data analyst may need to have knowledge of how to build visualizations. On the more technical side of data visualization that may include using a library such as D3 which requires some programming or it could mean knowing how to use BI reporting tools such as Business Objects or Sisense which don't require as much programming. To advance to a senior level data analyst they will need the mathematical skills to perform statistical analysis on the data and develop domain expertise their area of work."

Based on Mr. Mulkey's definition, data science is a combination of various aspects that we have seen reflected above. First and foremost is the ability to gather data from disparate sources, analyze this data, and finally, present the data to decision makers. He again emphasized strong communication and visualization skills. Mr. Mulkey continued beyond many of the previous sources by identifying core technical skills needed by an analyst. These skills include:

- Strong SQL and database knowledge
- Strong knowledge and understanding of NoSQL databases and the appropriate use for these tools
- Ability to use programming languages and other tools to extract, transform, and load data from these different data sources

- Strong mathematical skills to enable meaningful interpretation of the results of their analysis
- Finally, strong domain knowledge allowing the analyst to understand the problem and data they are working with

Mr. Travis Tucker, VP and Chief Analytics and Technology Officer for MiMedix Group, Inc. defined two major skills a graduate would need to enter a data analytics role and be productive. First, they must know how to get data out of different systems. He did not care if the data was captured using SQL or some other tool but he needs an analyst who can capture data from various sources. This could be accomplished using a programming language or it could be through the use of a tool such as SAS. The prospective analyst needs to be able capture unstructured data utilizing data mining skills. Second, the analyst must be able to structure a problem and solve it using proper mathematical reasoning in a step-wise manner. While machine learning is useful, it should be reserved for those problems that require it, not be the first avenue followed. Mr. Tucker wants an analyst to use the tools that are available and create new tools when these tools do not solve the problem. He felt the analyst needs a "bag of tools" to be successful allowing them to use the appropriate tool to solve the problem. As an analyst becomes more senior, Mr. Tucker felt that it was critical for them to be able to explain the problem and their analysis to decision makers.

Based on the industry papers and interviews conducted with local IT professionals, we believe the following skills are mandatory for a data scientist/analyst:

- Strong technical skills
    - Ability to utilize data intensive tools such as Hadoop or Spark

    - Ability to utilize provided tools such as SAS or SAP
    - Ability to utilize Machine Learning for data munging and extraction
- Strong communication skills
    - Oral and written communication
    - Visualizations
    - Ability to answer ad hoc questions
- Strong domain knowledge
    - Ability to interpret results and provide recommend conclusions in a specific domain
    - Ability to recognize valuable or misleading data to correctly guide decisions
- Strong mathematical and statistical skills

CHAPTER 6

An Example Heterogeneous Distributed Solution

This chapter uses significant portions of textual materials, graphs, and figures from Price et al. 2018 “Efficient Processing of Range Queries over Distributed Relational Databases” © 2018 IEEE International Conference on Information Reuse and Integration (IRI) [79]. Code supporting this research is shown in Appendix B.

6.1 Data Science Motivation

There have been several solutions proposed for solving the distributed range query problem. These involve specialized data structures that map the location of the various data elements. We propose a solution that solves the range query problem without the use of any auxiliary data structure. We have used this system to teach the heterogeneity of distributed systems to our students since they are familiar with Java and can see this execute disparate environments.

We discovered exposing students to a concrete example of a distributed system allowed them to grasp concepts related to middleware and centralized algorithms executing in a distributed manner much more easily.

We proposed a system allowing large datasets to be processed over distributed relational databases in an efficient manner. We did not attempt to address failure resilience in this system since other systems such as Consistent Hashing [80] and Chord [81] have been developed. While these systems address node failures, they do not provide robust query generation for range queries.

6.2 Experimental Motivation

This research addressed the problem of querying with a range of keys rather than querying using a single key point query. Range queries are an important aspect of data queries where a list of items needs to be generated. This list could be something as simple as a list of words beginning with the same letter or something as complex as list of products within a range of product IDs. Regardless of the domain, we routinely need to retrieve all items for this search.

Using ranges of keys introduces several complexities. First, all possible keys in the range must be generated. Since we do not know all of the key values stored, we must determine a method that will allow us to build this set of keys. If data is located on a single server or a system that utilizes proprietary technology that allows us to track items that have been stored, this simply becomes a matter of leveraging the built-in mechanisms such as a between clause in SQL. For a distributed system using a hashing algorithm to place data on remote servers, there is no mechanism in place to track what data has been stored. Therefore, we must generate every possible key to allow us to successfully retrieve data from these different servers. For short keys, this is not an expensive process and can be done quickly. However, for large key sizes, key generation takes a prohibitively long time to complete.

A further complication arose when the query was attempted. Sending the query to each server to retrieve possible data required a query for each key generated. This resulted in unacceptable performance for queries with a large number of possible keys.

The contribution of our research was to provide a solution for overcoming these two major problem areas and provide a robust system that supports range queries using

distributed relational databases without the need for special mapping tables or any other method of tracking data location. This has been accomplished for point queries, retrieving a single item from a distributed system, but this has not been accomplished for range queries. Our implementation uses distributed relational databases and does not require any of the remote databases to have any knowledge of other nodes. This system relies on a master server and any number of independent database server nodes. The master server maintains a list of remote database nodes and controls the routing of queries to these servers on an as needed basis.

6.3 Background and Related Work

Several systems have been proposed for implementing range queries using data structures to handle the range query. A range query allows a user to retrieve information between two given items. For example, a crossword enthusiast might be stumped and need a list of all of the words from a dictionary that begins with "ca". This could be done by executing a search on the dictionary requesting all words that begin with "ca" up to but not including words beginning with "cb". This retrieval would then allow the user to select a word to complete the puzzle they are solving. To create the query, the system would need to retrieve all words that begin with "ca" such as "caa", "cab", "cac", etc. Since we don't know each of the words that would meet this criteria, we would need to generate each possible combination of letters to see if it exists.

When performing a range query on a single system, this is easily done in a relational database. The challenge we encountered is we have no single source of all of the words. These words can be on any of the servers in our network. In this experiment, we used a hashing algorithm to distributed words across our servers and once sent to the

server, the master node immediately forgot all knowledge of where the word was located. Other solutions proposed to solve this problem have relied on data structures that maintain the location of this data. Some even proposed distributing like objects, those that begin or end with similar patterns, to specific nodes to eliminate the need to search across multiple servers to find the complete list of items. Our solution avoids bottlenecks and overloaded nodes that may result from this type of solution.

Ratnasamy et al "Range Queries over DHT's" [82] proposed the use of Prefix Hash Trees coupled with Distributed Hash Tables to form a trie-based scheme. In this paper, Ratnasamy et al, proposed storing the trie buckets in the DHT to allow for more efficient processing to find the corresponding prefixes. They developed the idea of placing the prefix hash tree in different buckets in the DHT. Comparisons were then done with generated keys to find the longest common prefix to use in the query. They proposed this could be parallelized by dividing a range into sub-ranges.

Yalagandula in "Solving Range Queries in a Distributed System" [83] referenced Skip Graphs, Aspnes et al [84] and Squid as proposed by Schmidt et al [85]. Skip Graphs rely on membership vectors as shown in Figure 6.1 which are tracked in a DHT. This DHT is built on the physical nodes. Each node has to maintain pointers to other nodes. Squid uses an n-dimensional data mapping to a 1-dimensional space. Squid is based on a data-lookup system that essentially implements an internet-scale distributed hash table. This system utilizes keywords to hold mapping. For example, numbers could be mapped in ranges of values and English words based on their first letter. This scheme is able to perform efficient range queries but the distributed loading becomes skewed.

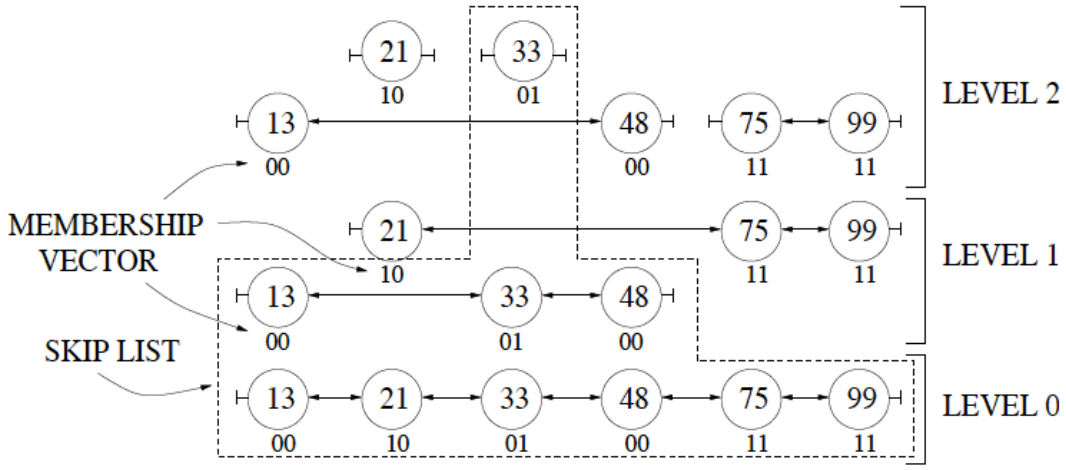

Figure 6.1: A skip graph with [logN] = 3 levels

In Yalagandula's approach, the alphabet is limited to one-dimensional keywords to allow for efficient distribution and lookup. The alphabet is limited and hashing is based on an ID which will correspond to the node storing the data element. In this approach, Yalagandula created tries with hash tables to map locations for queries. This is shown in Figure 6.2 for a tree of lower case letters.

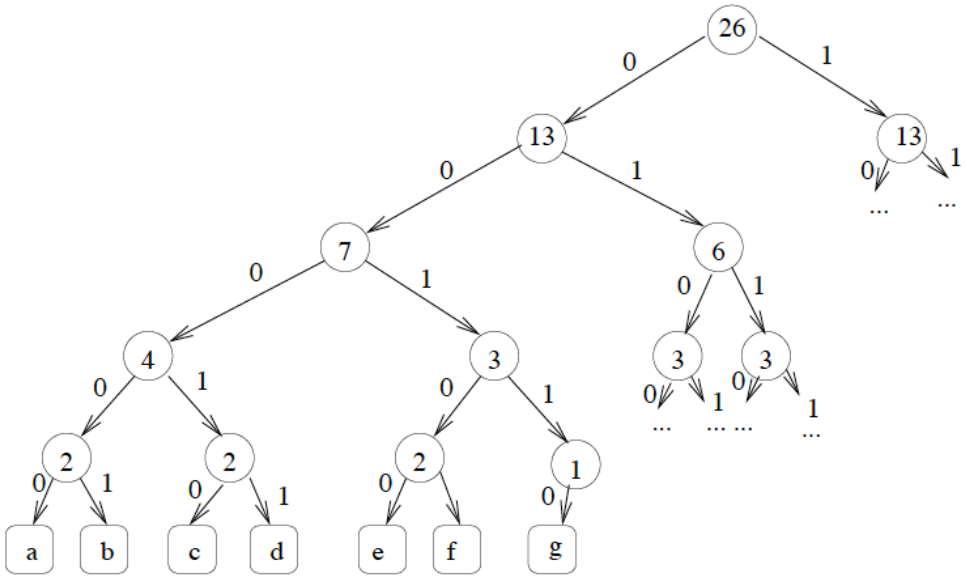

Figure 6.2: Encoding tree for representing 26 lower case alphabets

Once the Trie structure is created, buckets map the IDs to a node. When attempting to create fine resolution, this results in a very large number of buckets that need to be defined and maintained.

Our motivation for seeking a different range query approach was to remove the necessity of maintaining data on the master node and allow flexibility in storage and retrieval. This system is intended to be able to store any type of data that can be identified with a hashable ID. In this approach we differ from previous solutions in that no restrictions are applied to the keys or the alphabets used to hash and distribute the data. There is no need for any of the database nodes to know about other nodes. The master node must know about each of the slave nodes but we believe this can be handled with most distributed protocols.

6.4 Experiment Design

Our approach was to implement a distributed database using readily available technology, SQLite. This test was to process a standard English dictionary and store each word and its definition based on the hash of the word. We used varying key lengths and only used these keys to map the data. Once a hash was generated, this was used to map to the available servers.

To generate the list of keys, we employed a recursive, looping algorithm. This algorithm began with the initial word used in the range query and added characters to it until the desired key length was reached. The algorithm then generated all possible keys by incrementing each letter and adding all possible characters to the list of words to be queried. This was necessary to ensure that all servers that could contain a key, were queried. For example, if we use a key length of two characters, and we want to query for

all “words” between M and O, we would need to build keys for MA, MB, MC, …, MZ, N, NA, NB, …, O. Figure 6.3 shows the steps required for the key generation.

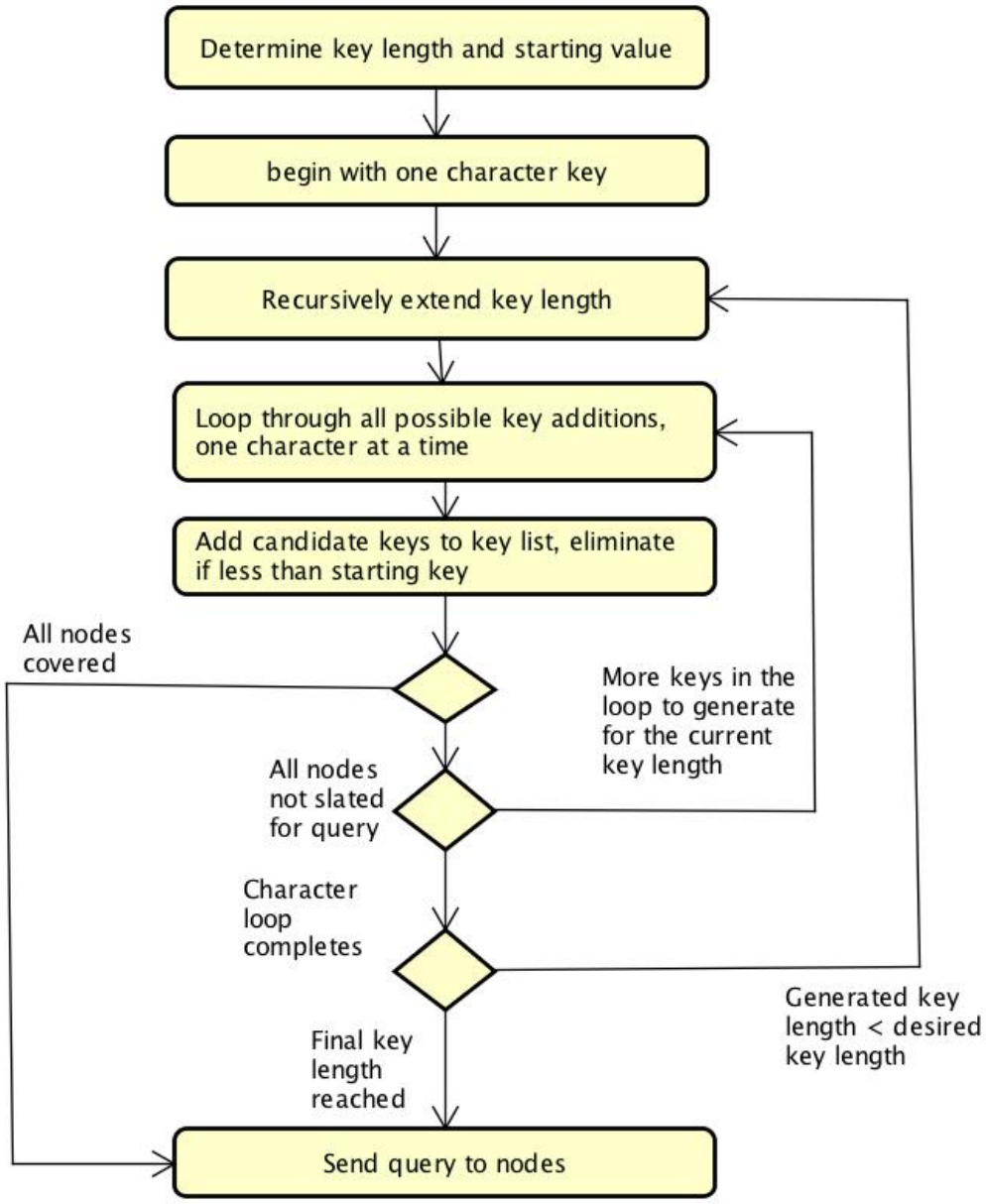

Figure 6.3: Key generation activity flow

Our initial attempt to solve this problem is illustrated in Figure 6.4.

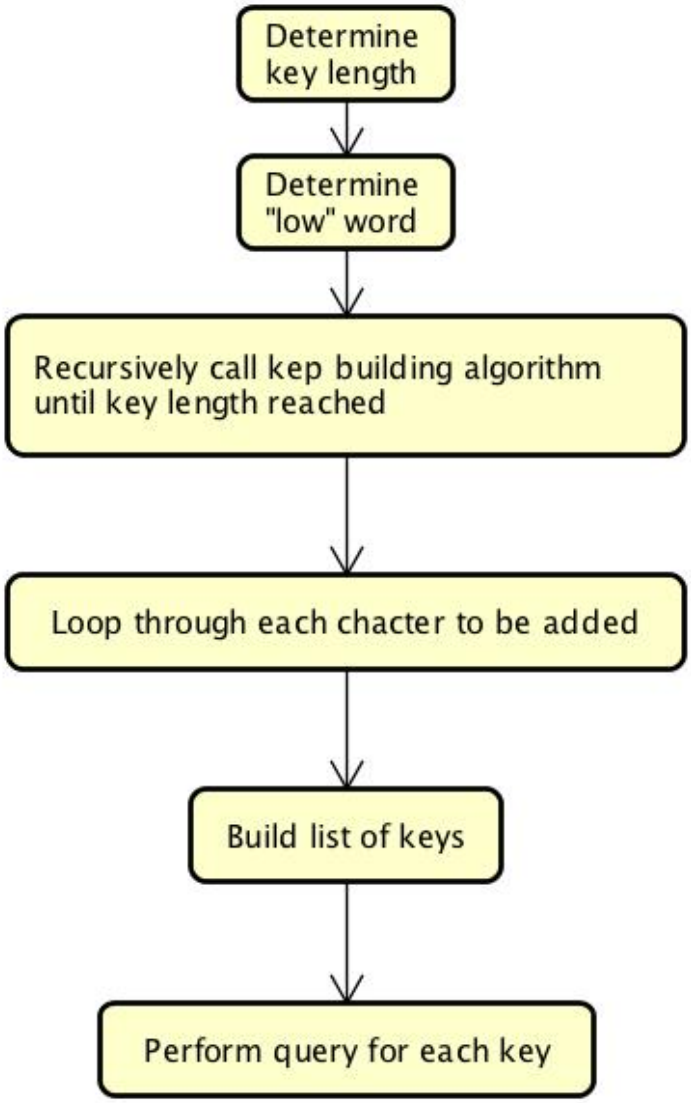

Figure 6.4: Initial program flow

In our initial experiments, we generated an exhaustive list of keys and sent individual queries to the servers containing the data. This proved to cause unacceptable delays for the retrieval of information due to the large number of queries being transmitted.

After interpreting the initial results, we modified the retrieval process to send a single query to each server that contained the data. This allowed us to retrieve all of the data from each server with only one query. It is important to note, servers that did not contain data were never queried. This optimization resulted in a dramatic reduction for retrieval, cutting retrieval times for “M” to “O” using four letter keys from 2,747,945 milliseconds to 235 milliseconds. Based on these results, we began examining other possible optimizations. Figure 6.5 shows the revised program flow for this communication improvement.

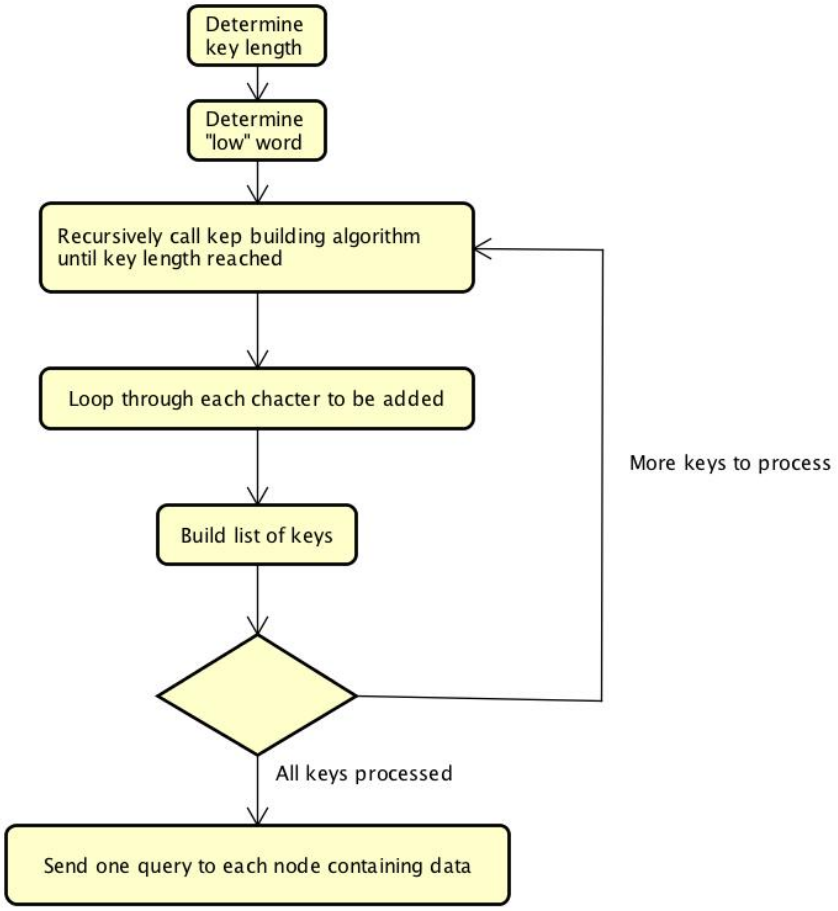

Figure 6.5: Query optimized program flow

When keys grew beyond five characters, the generation of all possible keys became another bottleneck. This was the final major bottleneck we encountered. Assuming the key set is limited to the letters from the exclamation point(!) to the tilde symbol (~), for every position in the key you must traverse for each of the subsequent possible key additions, 95 keys are generated at the first level, this quickly becomes an extremely large number of keys. For example, with a key length of six, to generate all of the keys for "A" would be 1 for the first letter, 95 at the second level. At the third key, it would be 1 + 95 + 95 squared. At the fourth level, it would be 1 + 95 + 95 squared + 95 cubed. Roughly, we have $95^{key\ length-1}$ keys. For a single key with a key length of 10, this generates $6.302\ x\ 10^{17}$ keys. The generation of these keys resulted in unacceptable performance.

6.5 Experimental Breakthrough

When considering these issues, we discovered tracking the servers needed was the optimal solution. Without this optimization long keys require extremely intense calculations with unacceptable performance. By tracking the number of nodes available and the number of generated keys, the key generation can be done very rapidly. Once the number of servers to query equals the number of servers containing the data, there is no reason to continue generating keys. As shown in Figure 6.6, we allowed the key generation to short circuit and exit when all possible servers were in the query queue. The key generation algorithm rapidly exits and returns the proper results.

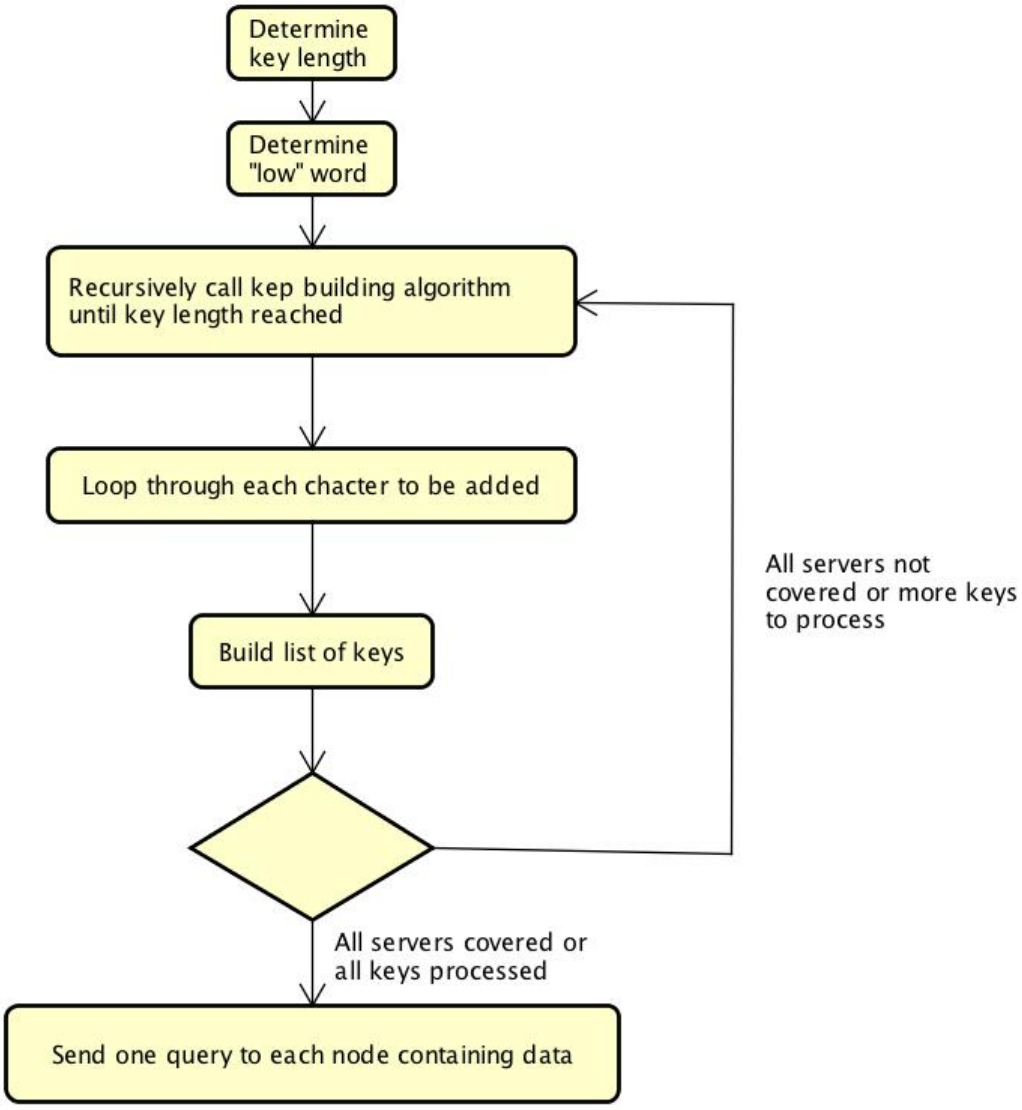

Figure 6.6: Final range query program flow

6.6 Experimental Results

In our experiment, we used 30 Ubuntu virtual machines with 1 GB of Ram and a single CPU. These were hosted on servers in our networking laboratory and resided on multiple racks. The master node was a Macintosh 15-inch laptop with 16 GB of RAM, a 500GB SSD and quad core I-7 2.5 Ghz processors. We emulated 100,000 nodes and then

mapped the queries to these 30 remote devices. Communication was done over the internet with an average ping time of 20 ms. Timing was performed using the milliseconds of the master clock and was reported from the point where the query began, including key generation, communication to each node and the processing/compilation of the results from each query.

Not surprisingly, all of our implementations worked efficiently on short keys, less than four characters. A one-character key resulted in an uneven loading across 93 servers, the data used was an English dictionary, the remaining servers were completely ignored. A two or three-character key resulted in more balanced loading across the servers and reasonable performance.

To solve the problem of which servers to query using a range query, an exhaustive list of keys needs to be generated. This requires a complex recursive algorithm to generate each possible key. This algorithm combines a recursive component to build the keys to the length required and an internal loop to add every possible set of keys. While our implementation efficiently generated all keys for queries using keys less than six characters in length, when generating a key eight characters long or greater, extensive computation was required and there were unacceptable performance delays in the query. If a starting or ending filter was shorter than the key length, the filter was extended so that it was at least as long as the key length specified. For the starting filter, a ! character was added until the filter was the correct length. For the end filter, a ~ was added until the key length was achieved.

An additional problem was the number of queries sent to the servers. Originally, we were sending a query for each key generated. This flooded the network with queries and resulted in extremely poor performance.

Figure 6.7 shows the time required to send each of these queries and retrieve and collate the retrieved data. Due to the extreme times required to execute queries using the original plan, only key lengths of one to four characters are shown.

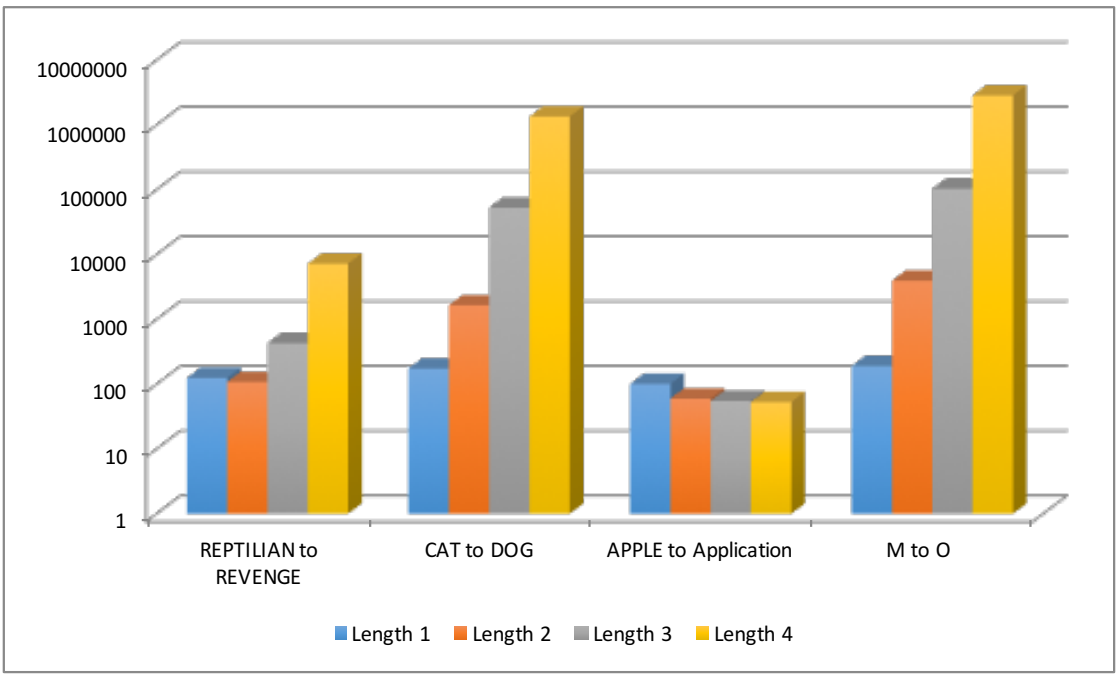

Figure 6.7: Time to retrieve and collate data based on key length in ms (log scale)

This led us to two changes in our process. First, we cached the queries and only sent a single query to each server. This greatly reduced network traffic. For example, this change reduced the number of queries for a key length of four from 52,728 queries for the filters “M” to “O” to no more than the number of servers. The second change was to short-circuit the key generation and only build keys until either the number of servers was

reached or all the keys for a given filter set were generated. Even when the number of servers was increased dramatically, the key building remained efficient. In our experiments, generating enough keys to hash to 100,000 servers required less than 500 ms. Figure 6.8 shows the key generation times based on our revised algorithm. After implementing these changes, execution times were greatly improved.

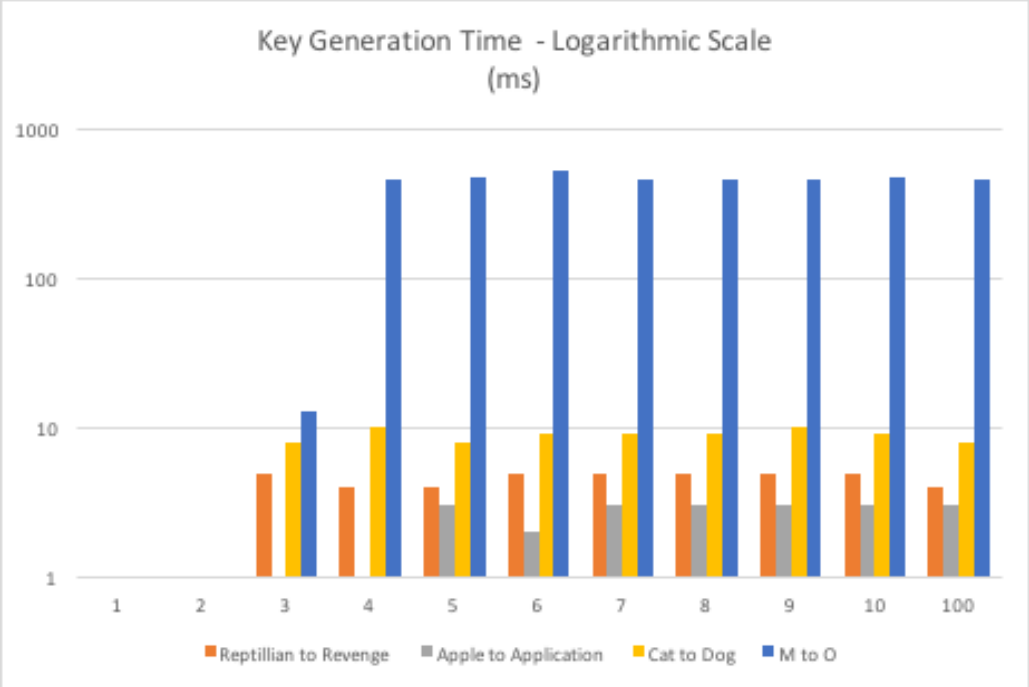

Figure 6.8: Key generation times in ms (log scale)

Our experiments were performed using 30 database servers with distributions of keys over 100,000 virtual servers. Each of these servers was remotely located and accessed via the internet. Figure 6.9 shows the times for retrieval for varying key lengths.

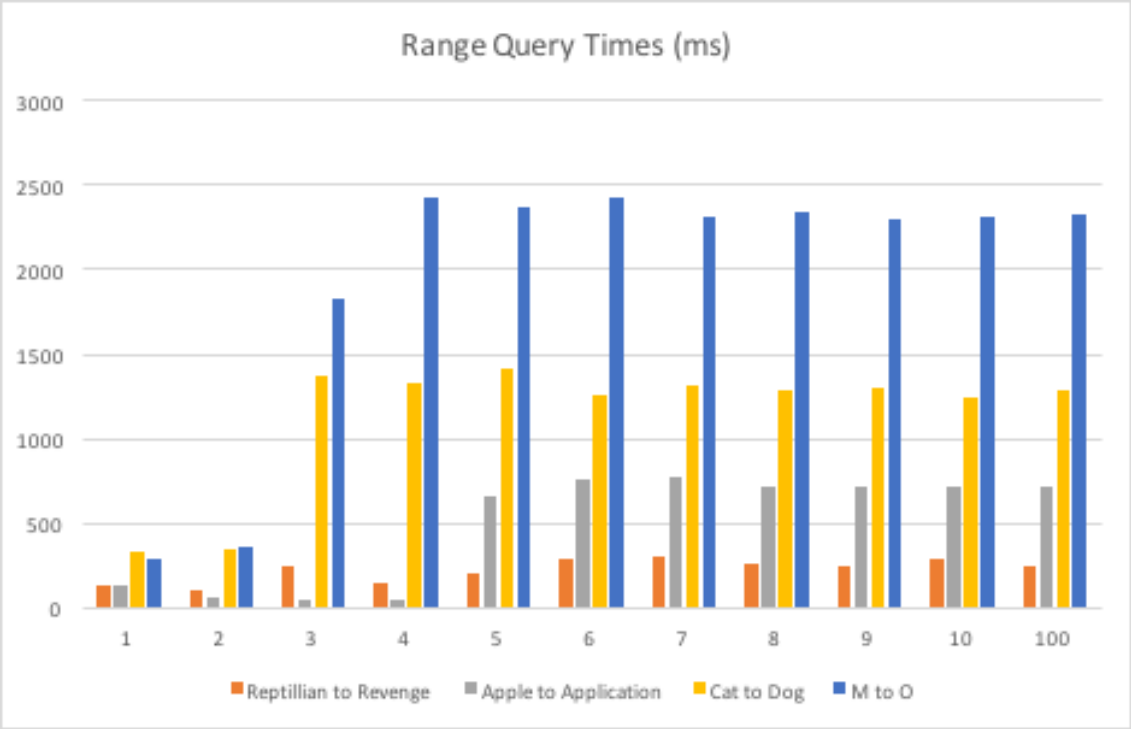

Figure 6.9: Range query times in ms

By tracking the number of servers that the algorithm returned, we can see that this number stabilizes when the word is shorter than the key length. Figure 6.10 shows the number of servers the algorithm returned. Even when returning 100,000 servers, the algorithm performed in less than 500 milliseconds. By using 100,000 servers, we believe we have modeled an extreme case with 100,000 servers to test the short-circuiting theory. Based on these results, we believe tracking the servers and stopping key generation when further generation cannot add additional servers is an effective way to generate keys for range queries.

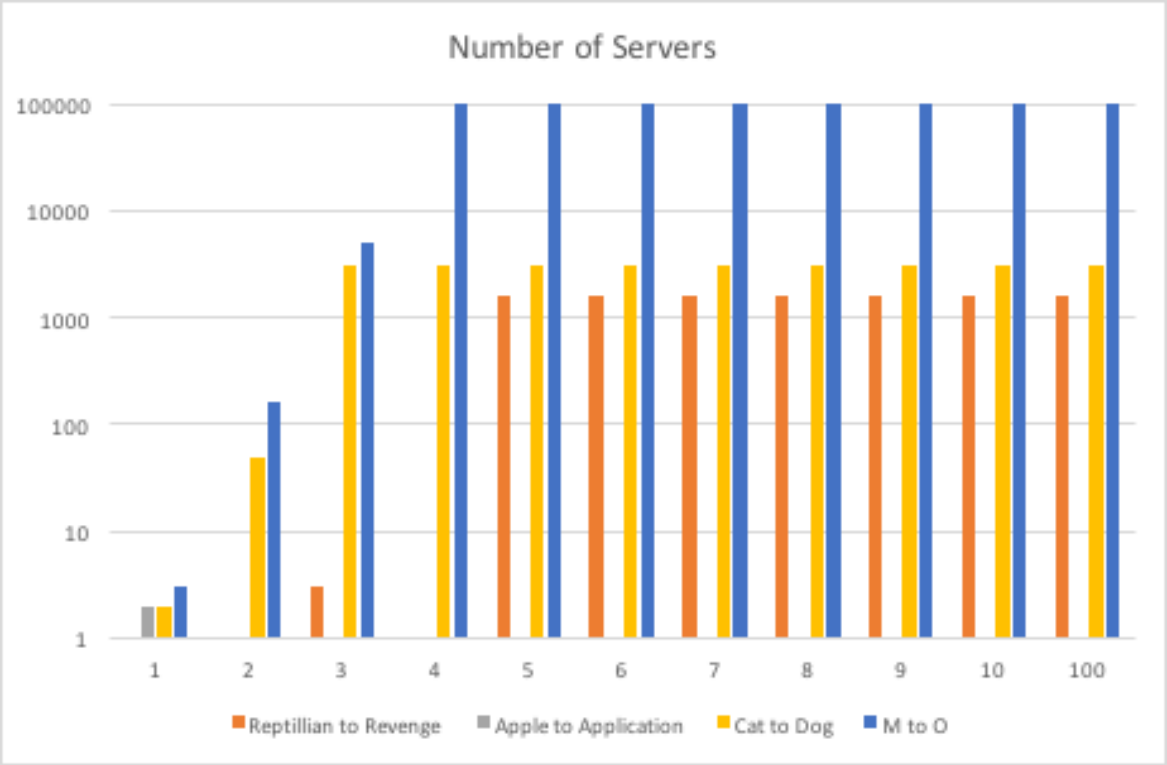

Figure 6.10: Number of servers generated using different key lengths (log scale)

As we increased the size of the key, we saw a dramatic decrease in words returned. This was the result expected. By increasing the "filter", the return data set became more precise. Figure 6.11 shows the result of the queries with different key lengths.

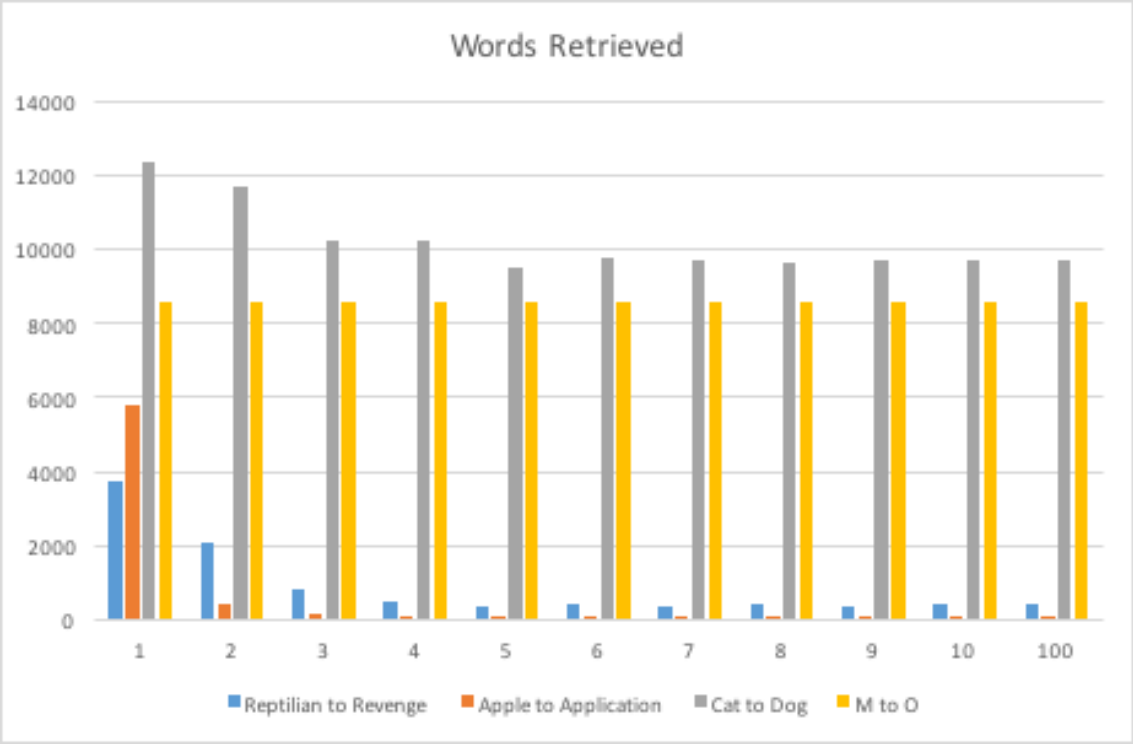

Figure 6.11: Number of words retrieve using different key lengths

6.7 Conclusions

Range querying can be efficiently performed. An exhaustive set of keys must be generated and hashed to determine the correct servers to query. For longer keys which result in more accurate results, this can be computationally prohibitive in terms of time and performance. Likewise, sending a single query to each server can be done but again this quickly becomes non-performant. By short-circuiting the key generation once all of the servers to be queried are included and sending a single query to each effective server, performance can be optimized. This method of distributed querying should be adaptable to any data that has hashable keys.

By exposing students to a complex problem such as this and showing them how we can ignore different operating environments, we were able to more easily teach

distributed middleware and eventually the entire distributed system concepts that are crucial to data science processing. We believe any system that mirrors the distributed nature of this application can be used to help students understand these concepts.

CHAPTER 7

A Framework for Introduction to Data Science for Information Technology Students

This chapter uses significant portions of textual materials, graphs, and figures from Price et al. 2019 "**C**hallenges and Approaches to Teaching Data Science Technologies in an Information Technology Program with Non-traditional Students." [86]

7.1 Introduction

Teaching Data Science to traditional Computer Science majors presents many challenges to students, faculty and the supporting infrastructure. When the student population changes from Computer Science students to those in an Information Technology program where many of the core Computer Science courses are not required, additional challenges arise. Couple these challenges with teaching non-traditional students who are also involved with families and careers; the difficulty in teaching this material rises exponentially. For our experiment, we chose to teach the Hadoop Distributed File System, the Map/Reduce paradigm, and basic Hadoop programming to expose these Information Technology students to Data Science technologies. In our recent experience teaching Data Science to Information Technology students we faced many challenges that we had to overcome, including missing distributed systems knowledge, a lack of basic Unix and operating systems understanding and a complete unfamiliarity with Distributed File Systems. In addition, these students came from very diverse backgrounds including many different ethnicities and both male and female

students. We were able to measure the effectiveness of teaching prerequisite material using control and experimental sections and will discuss the impact the prerequisite material had on student success.

Students in this study were senior Information Technology students in their capstone course. Each of these students had previously completed three programming courses; two courses in software engineering, and a database course. Other required courses these students typically have completed prior to this course include statistics, calculus, networks and a system analysis and design course. There are numerous electives that may have been completed prior to this course including artificial intelligence, game development, mobile development, and web development.

With the current need for data scientists, we realized the necessity for our students to have some exposure to data science technologies. With a predicted shortfall of data scientist trained personnel of between 140,00 and 190,000 for 2018, we felt it was critical to expose students to some of the data analytic toolsets. Song, TREND and Tang et al. identified that 90% of the data in the world today has been created in the last two years. [87] [88] [89] [90] Tang et al. state approximately 1,200 exabytes of data are produced annually. [89] With the predicted shortfall of qualified personnel and explosive growth in data, we felt our students had to have a basic understanding and interaction with data-intensive computing.

We selected Hadoop for multiple reasons. First, its popularity and power to process large data volumes as well as being a customizable platform that can be adapted to many data analytic problems. Second, the availability of resources to help students

learn the basic technology and how to apply it. Current work is leveraging Spark and we intend to add additional technologies as time permits.

Our students are enrolled in an Information Technology degree rather than a Computer Science degree. This presented unexpected challenges because some of the core courses in a Computer Science program such as Operating Systems and Distributed Systems are not required for this major. In addition, these students are primarily commuters with many non-traditional students most of whom have families and careers not in the information technology field.

We will discuss the challenges faced and solutions used to teach Hadoop setup, programming and interpretation to this group of students. Classes were face to face and consisted of a mix of traditional and non-traditional students.

When we began this experiment, we believed that students possessed the prerequisite knowledge to proceed directly to the data intensive computing portion of the course. Based on the struggles our students encountered, we modified our pedagogy to teach basic concepts in operating systems and distributed systems. Since these are not required courses for these students, many of them had no experience especially in the distributed systems area. Students in this experiment were not Data Science track majors but Software Development students. This material was added to their capstone course to provide Data Science exposure requiring us to complete all aspects of this project as well as the software development related material in one semester.

7.2 Related Work

Ramamurthy identified 6 courses that were required for teaching Data Science in "A Practical and Sustainable Model for Learning and Teaching Data Science". [58] One

of these courses is Distributed Systems which would have greatly reduced complexity in this project.

In "An Undergraduate Degree in Data Science: Curriculum and a Decade of Implementation Experience", Anderson et al. identified core Data Science, Computer Science and Mathematics courses for an undergraduate degree in Data Science. [67]

7.3 Experiment Design

We took a two-prong approach to solve the challenge of teaching data science concepts to Information Technology students. First, we taught the course and using observations, surveys and feedback from students determined the areas where the students struggled. After gathering this data, we formulated our hypothesis that teaching critical components not necessarily related to data science first would improve student understanding and ability to master the data science material. The three areas we selected were basic Unix operations and understanding, distributed systems foundations and finally the Hadoop Distributed File System.

Before we taught this course, we prepared the material we felt was necessary to teach students this technology and its use. Originally, we expected about three weeks would be required to cover the concepts and allow the students to program a simple Hadoop program mapping outgoing unique links from a data file containing over 10,000,000 entries.

Our plan began with preparation of installation, programming, and analysis guides. The original plan for teaching the Hadoop concepts is shown in Figure 7.1.

We elected to administer a post survey to collect feedback from students to confirm our observations.

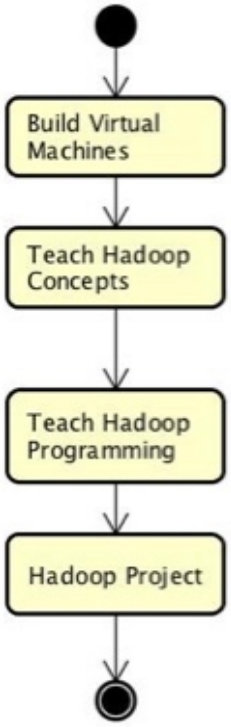

Figure 7.1: Original Teaching Plan

The original experiment was simply to record the challenges unique to non-traditional students when they were confronted with problems that required solutions using different paradigms than those to which they already had exposure.

As the experiment progressed, we began designing the prerequisite material to allow an easier transition. To validate this portion of the experiment, we introduced pre-tests and post-tests to measure the effectiveness of the prerequisite material as well as

refining the projects to measure student understanding. We were fortunate to have both controlled and experimental sections to confirm the validity of our hypothesis.

7.4 Challenges

Shortly after we began, we realized some core knowledge had not been taught to these students and we needed to help these students understand the concepts behind this technology. This core knowledge included operating systems concepts; remote method invocation and communication procedures; and basic distributed systems concepts, especially distributed file system, concepts. For many of these students there was a complete unfamiliarity with Unix based systems both from the permissions required and the command line interface.

The operating environment chosen for this experiment was a virtual machine install of Ubuntu 16.04.3 using Hadoop 2.8.1 and Java 1.8.0.144. Different virtualization tools were used including Virtual Box and different flavors of VMWare's tools, VMWare Player and VMWare Fusion. The choice of virtualization tool was left up to the individual student. Allowing the students to use their choice of tools created some significant obstacles.

7.4.1 Installation

Creation of operating environments proved to be a significant challenge. For most of these students, this was their first exposure to Unix. While many of them had used virtualization software, others had not had any exposure, therefore, we had significant challenges related to different tool choices. Some of the challenges from the virtualization tools resulted from differences in basic functionality, connecting to the host

machine, connecting to USB devices, and network access. In addition to the virtualization challenges, Unix itself proved to be an unexpected hurdle. Students were not familiar with the command line interface, especially in a Unix environment. Sudo was a foreign concept to many as well as the ability to create symbolic links. This required us to explain and to provide training on core Operating Systems concepts.

Java installation proved to be a minor challenge once the operating systems hurdles were overcome. Several students had problems with this installation but most of these were easily overcome by walking through the install with the student and explaining the different steps in the procedures supplied. This did expose a recurrent problem with the students' willingness/ability to follow the provided setup procedures. This became a reoccurring theme throughout this experiment. Many of these students did not follow the step by step instructions provided either due to a lack of knowledge or a disregard for the provided instructions.

Hadoop installation proved to be a tremendous challenge. The Hadoop binaries were shipped as 32-bit binary and had to be rebuilt to work correctly without error in a 64-bit environment. This required installation and training on many different build tools including Maven, libssl-dev, build-essential, pkgconf, cmake and libprotobuf9v5 protobuf-compiler. While these tools were easily installed on Unix, the proper configuration of these tools was a challenge. The POM file supplied with the Hadoop source specified incompatible versions of some of these tools and had to be modified. Once these modifications were made, the build itself simply took time to complete.

Our post-experiment survey showed students felt installation of the tools on the Unix operating system was difficult with 63% of the students rating this portion of the

experiment as moderately or significantly difficult. Data from the student survey is shown in Figure 7.2.

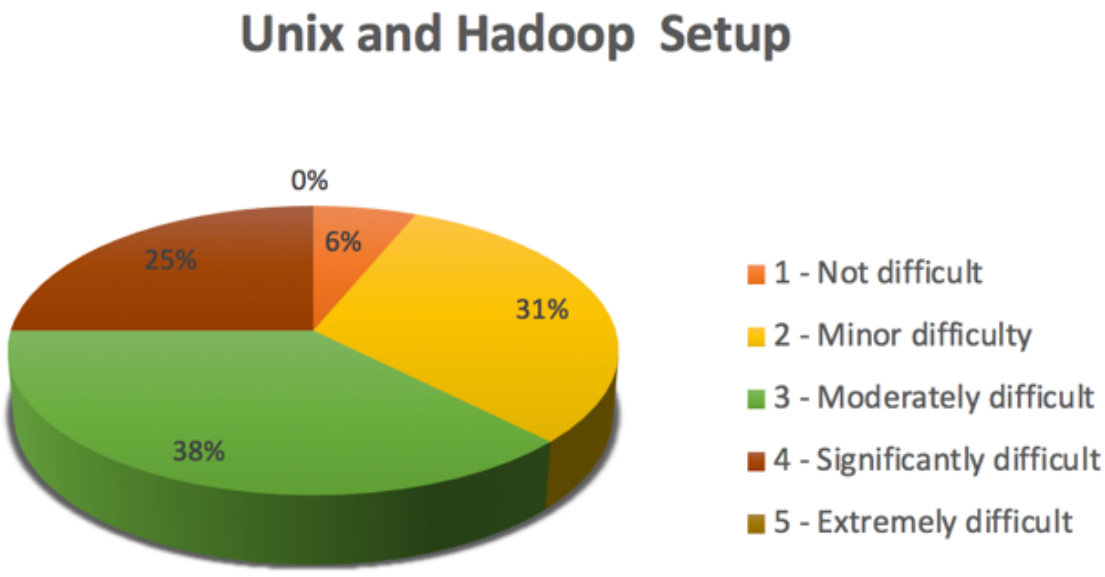

Figure 7.2: Unix and Hadoop Setup Difficulty

7.4.2 Configuration

Hadoop configuration after installation required many changes to the supplied XML files. While Apache documented most of the changes required, there was one major problem not addressed in the Apache documentation which caused the Hadoop Distributed File System (HDFS) to be erased every time it was shut down. The hdfs-site.xml file had to be modified to provide a permanent location for the HDFS. Without this modification, the HDFS was erased every time the HDFS was terminated. While students had been exposed to XML files in earlier courses, many of them were uncomfortable navigating to these files and modifying them. Many of these files were

located in disk locations that required administrative privileges to modify. This became a stumbling point for many of these students.

Configuration of Hadoop was easy for some of the students but proved to be a challenge for others. This was a surprising revelation since the students all began with the same configurations and had the same instructions to follow. Over 56% of our students felt that configuration was moderately or significantly challenging. Figure 7.3 shows the percentages of the student ratings for difficulty with configuration.

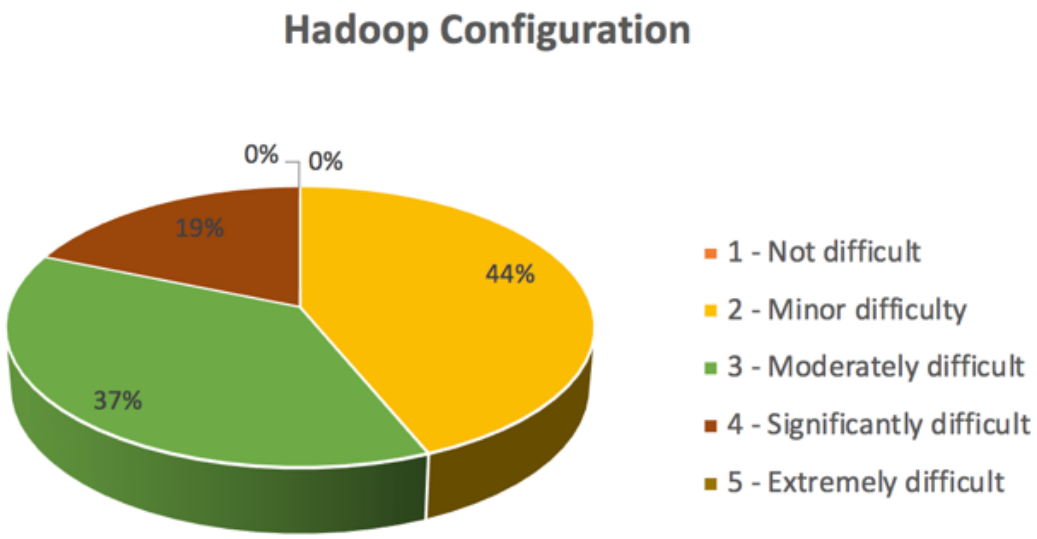

Figure 7.3: Hadoop Configuration challenges

7.4.3 Machine Capabilities

Since the students needed to run this system in the virtual environment on their own machines, we ran into problems with the capabilities of these machines. Running Hadoop in a virtual environment put a strain on many of these machines both in processing and disk space. This caused performance and installation issues.

Students did feel their machine's performance had a significant impact on their ability to learn Hadoop. Figure 7.4 shows 50% of the students felt this had a moderate or higher impact on their learning.

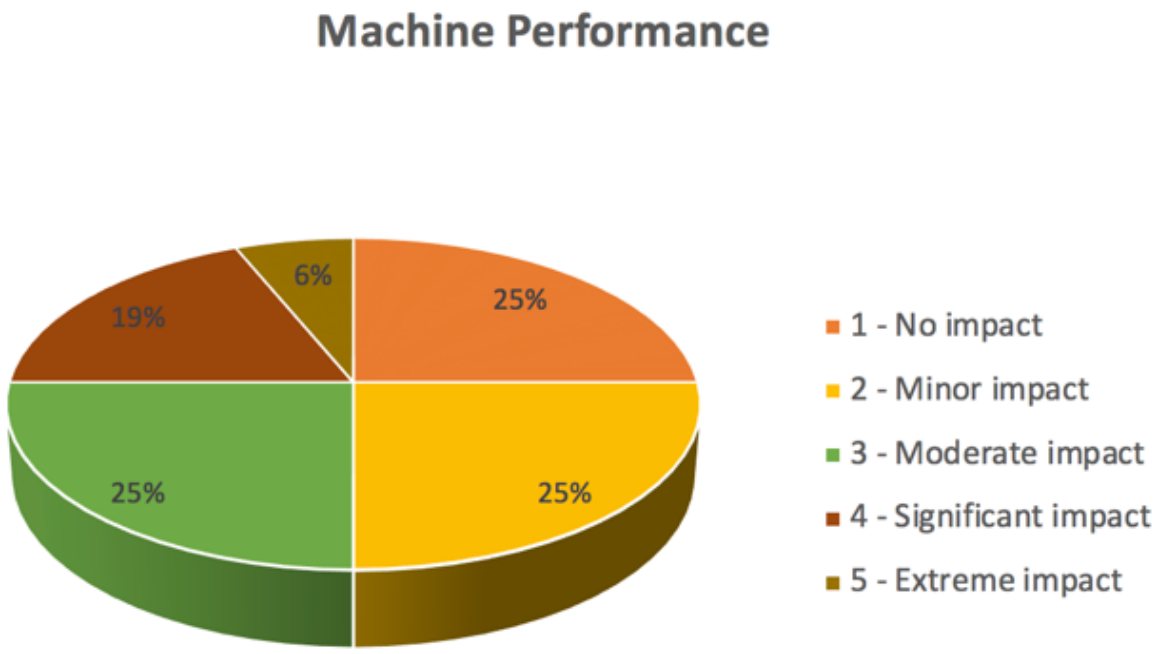

Figure 7.4: Impact of Machine Performance on Hadoop Learning

7.4.4 Distributed Systems Knowledge

Teaching the Hadoop system, especially the Hadoop Distributed File System was challenging since most of these students did not have a solid base in distributed systems. Their knowledge of heterogeneous systems relying on middleware was weak. Without this knowledge, the concept of a distributed file system was hard for them to understand. This made navigation on the file system difficult. Students had a difficult time using command line commands to navigate and process files within the command line environment.

Having never taken a formal distributed systems course proved to be an extremely difficult barrier to overcome. 53% of the students rated this as a moderately difficult or higher while the other 47% rated it as a minor difficulty. Figure 7.5 shows the difficulty breakdown caused by a lack of distributed systems knowledge.

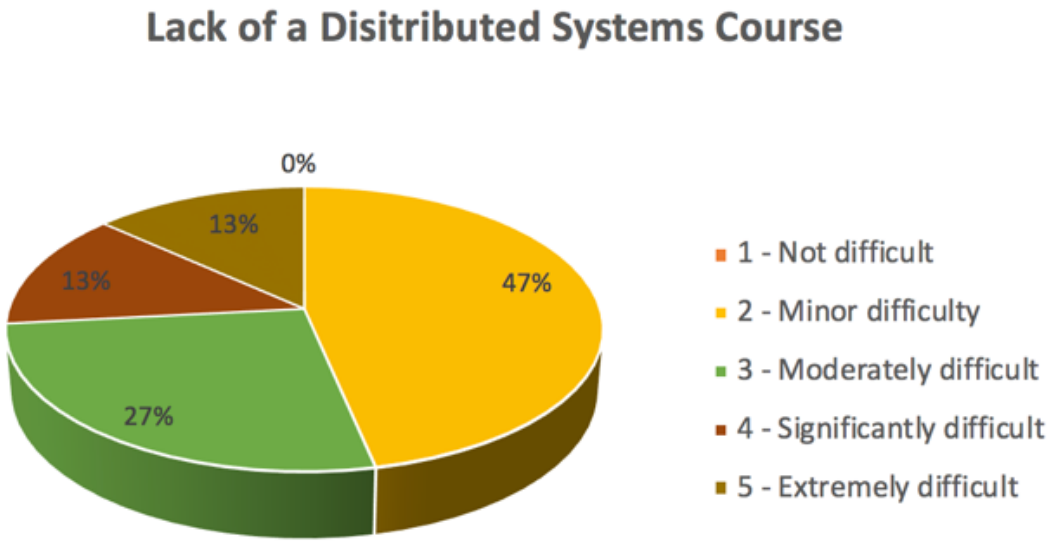

Figure 7.5: Impact of No Distributed Systems Course

7.4.5 Map/Reduce Paradigm

While we knew this was a new paradigm for the students, we did not expect it to pose the level of difficulty that we encountered. Students failed to understand the necessity for the Map to complete prior to beginning the Reduce due to the key separation imposed in Map/Reduce. This proved to be an enormous stumbling block when we began programming Hadoop jobs. Many of the students believed the Reduce would start prior to the Map completing and therefore wrote Reduce code in the mappers.

After teaching students these different concepts, programming constructs in Hadoop were easier for them to understand. During the programming process, we allowed them to have several failures with Hadoop programming to learn this paradigm. After completing the guided programs, they did not consider this to be a significant challenge. Figure 7.6 shows more than 50% of the students thought this was a minor or lower difficulty.

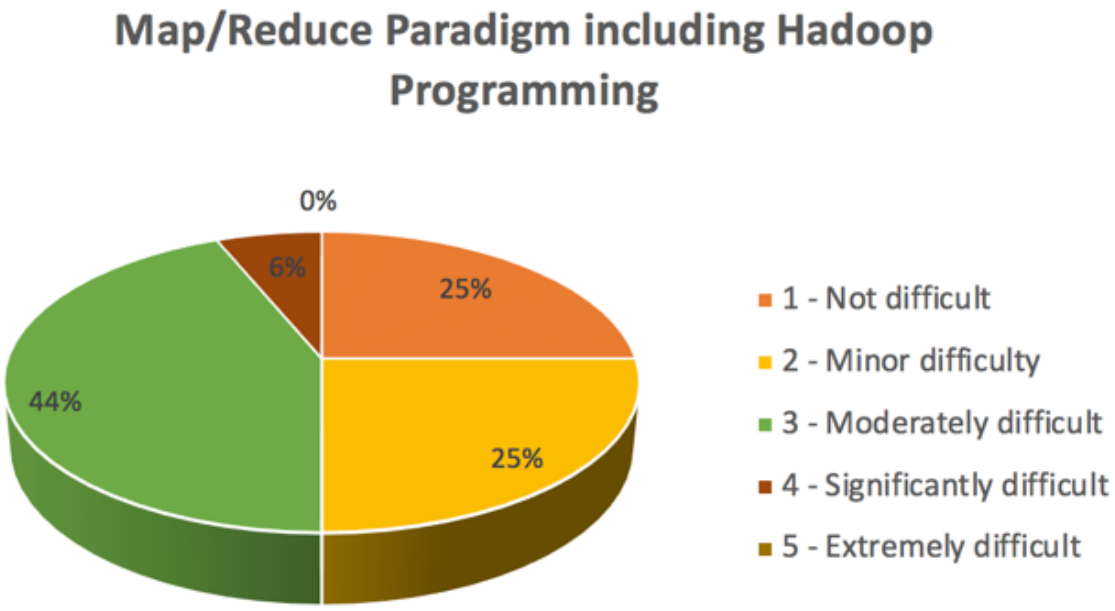

Figure 7.6: Difficulty Learning the Map/Reduce Paradigm

7.4.6 Hadoop Concepts in General

Because of the lack of understanding of distributed systems, the Hadoop Distributed File System was a foreign concept. This lack of distributed systems knowledge also raised a challenge in understanding how the Name Node and Data Nodes interact. The setup of the HDFS and how distributed processing occurred on the system was a challenge throughout this evolution.

The Hadoop programming interface proved more difficult to understand than expected. The main problem encountered was the lack of understanding of the programming paradigm associated with the key/value pairing and the distributed nature of this system. Another programming issue was the multiple places where the input/output data types had to be declared and matched. Many of our students struggled with the different data types imposed by Hadoop and the multiple places these needed to be declared and matched.

The final key concept that proved to be difficult was the semi-structured form of the data and how these could be mapped to the key/value pairs.

Again, student surveys showed this to be a difficult challenge for some but for many of our students, this proved to be a minor issue as shown in Figure 7.7.

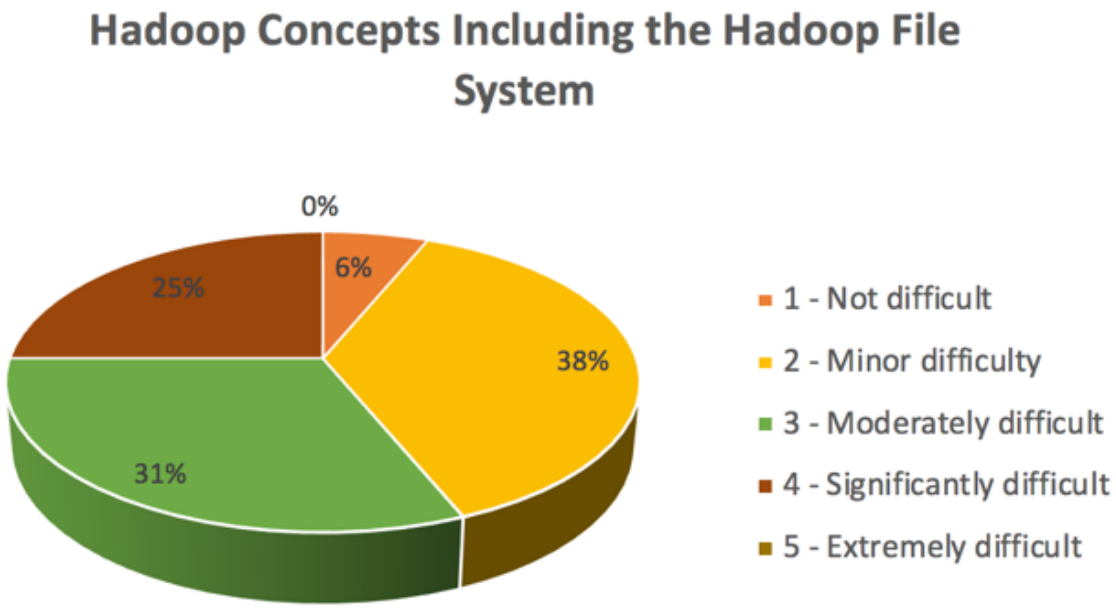

Figure 7.7: Hadoop Concepts Including the Hadoop File System

7.4.7 Family/Career Challenges

Many of these students were non-traditional students. These students have careers and families to support requiring large time commitments other than their studies. The demands on their time made it difficult for the students to spend the time in a traditional manner necessary to master these concepts. This dynamic also made it more difficult for many of them to seek help outside of class time. Students held many incorrect concepts when they were first exposed to this technology including a failure to understand the Map/Reduce paradigm, lack of understanding of distributed systems, and command line execution and processing. These knowledge gaps included a lack of understanding of jar file execution and the knowledge of how to apply and interact with external libraries/programs, especially from the command line. These and other areas where the foundational material had not been previously provided proved to be difficult for the students to overcome.

For many of the students in this experiment, the online chat sessions and late-night group chats we used proved to be invaluable. This allowed them to seek help when they could find the time in their demanding schedules.

As shown in Figure 7.8, family and career challenges were viewed very differently by this group of students. This did not come as a surprise since some of these students were traditional full-time students while others were non-traditional learners. Responses to this survey question were distributed across the entire spectrum of possibilities.

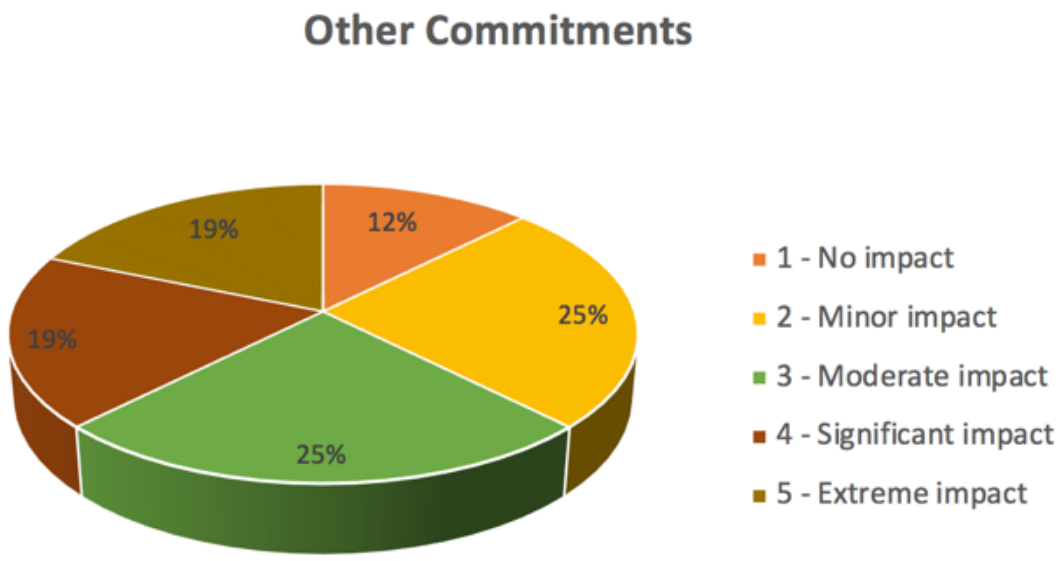

Figure 7.8: Family and Career Challenges

7.5 Experimental Revision

After beginning to teach this material, we discovered we needed to teach some prerequisite material to fill the gaps in student knowledge and allow them to master this material. After observing the struggles our students were having with Hadoop concepts, we went back and taught some very basic concepts in the Unix operating system, Distributed Systems and then retaught the Hadoop Distributed File System. After teaching this material, we observed a much higher success rate for both traditional and non-traditional students. This led us to the second phase of our experiment where we taught experimental and control sections for this course. In this portion of the experiment, we decided to teach material covering the basics of Unix, Operating Systems, remote method invocation technologies, and Distributed Systems. We then taught the different Hadoop concepts including the HDFS, Hadoop interface and Hadoop programming. We measured the success of this material by using pre- and post-tests. The pre-test scores

showed that our sections possessed similar knowledge of the core and Hadoop concepts. We then used the post-test scores and project success to determine if the material we had taught had any impact on the student's ability to master this material. Section 7.7 discusses the effectiveness of our core knowledge teaching.

7.6 Solutions

7.6.1 Installation

Although we created detailed steps for installation and virtualization of the operating system, we had to spend significant time assisting students install Unix, Java and Hadoop. We spent class time walking through the install. Even having done this, we still had to help some of the students individually with their setup and installation for Unix, Java, and Hadoop.

We created the 64-bit binaries eliminating the need for students to build them. Even this proved to be a challenge with the different virtualization tools that were used. Some students were easily able to download and install the Hadoop binaries while others needed help working through the limitations of their virtualization tools. After doing research on these tools, we were able to overcome these difficulties and to get all students setup with a working environment.

As this experiment progressed, we built virtual machines on our network servers and had students connect to these pre-built machines. We did provide detail setup instructions and assisted those students who wanted to pursue this alternative. The result was a much smoother operation and avoided many of the machine limitation problems we encountered when students were executing on their own computers. This did expose

another problem; the lack of familiarity with SSH and SFTP which were require to use these servers.

7.6.2 Configuration

To overcome configuration problems, we created detailed instructions and supplied these to the students. We worked with each student, as needed, to ensure configurations were set correctly. We also discovered during this time that some students either would not or did not know how to follow instructions. This required us to assist them in learning to follow the setup and configuration instructions. We also taught XML file formats and rules as well as Unix OS commands. By the end of this project, students were comfortable navigating on the command line and modifying files in the Unix environment.

The configuration problems were eliminated when we moved to pre-built virtual machines for those students using that solution. For students who decided to build the environment, this sometimes was a challenge but was not a limiting factor in our teaching of data science concepts.

7.6.3 Machine Capabilities

While there was not much we could do to solve these problems, we helped students optimize their machines and eliminate unnecessary bottle-necks for Hadoop processing. Some of the steps that were followed included providing guidance to free machine space and stopping unnecessary processes while the virtual machines were running. The provided pre-built virtual machines running on a college supplied Unix server eliminated many of the machine limitation problems. Unfortunately, we then encountered connectivity issues and the previously mentioned gaps in student knowledge

such as the use of SSH and SFTP which were needed to allow students to connect and move files to and from these virtual machines.

7.6.4 Distributed Systems Knowledge

Overcoming the lack of distributed systems knowledge was a larger challenge than the previously discussed challenges. To help the students understand these concepts we had to teach the basics of distributed systems. We began by teaching the definitions and characteristics of distributed systems. The concept of heterogeneous operating systems working through a middleware to handle distributed applications provided students with the framework necessary to understand how the HDFS works. We introduced the Range Query program discussed in Chapter 6 to provide an example of a system running in a heterogenous environment leveraging a common middleware. While this is not truly a distributed system, the familiar concepts of using Java and experiencing it run across multiple operating systems greatly facilitated the student understanding of a middleware process. By teaching the concepts of distributed algorithm processing, students were more rapidly able to grasp how the Hadoop system and the HDFS work together to process large volumes of data. This also helped them understand Hadoop failure resilience and processing increasing the scalability of the Hadoop process.

Once our students understood these basic concepts, teaching the Hadoop File System and Hadoop processing was a much simpler undertaking. By helping our students understand these basic concepts, the Map/Reduce paradigm was easier to explain. This allowed us to explain the distributed nature of Map/Reduce and how Hadoop leverages the HDFS to process information.

7.6.5 Map Reduce Paradigm

In our teaching demonstration programs, we created output that allowed students to see how Hadoop processes the Reduce based on the key. Once students grasped how Map/Reduce works on key/value pairs, many of the confusing points in their programming were eliminated. The example we used printed the keys and values as they were being processed by the Reduce. This helped students understand how keys are processed and mapped in the Mapper and then fed to the Reducer grouped by key. When we discussed this, one of the key revelations for our students was that the Map and Reduce processes can be run across multiple machines. These examples also helped our students understand that the map process must complete before the Reduce process can begin.

7.6.6 Hadoop Concepts in General

After addressing the lack of distributed systems knowledge, the Name Node and Data Node concepts were much easier for the students to learn. They still had challenges understanding the functions of these two nodes and how they interact to allow the Map/Reduce paradigm to proceed. Relying on the distributed systems knowledge we had taught, these concepts were much easier to explain.

The programming interface difficulties were overcome, for the most part, by simply explaining the different classes and having the students utilize the Hadoop API. One of the best techniques we used to overcome these difficulties was to have students research Hadoop, Hadoop programming, and the Map/Reduce paradigm and present their findings to their class. This allowed us to have student perspectives on this paradigm and

allowed us to clarify many of these concepts. This also encouraged the students to do more research to further their own understanding.

Our students were used to processing data contained in a relational database but had not used semi/un-structured data before. We discussed how to identify and map data into key/value pairs. Basic data mining techniques were taught to help students understand different ways of processing this data. We provided examples of mappings using relational database terms which helped solidify their understanding.

In our revised teaching plan, we provided this basic information before trying to delve into the distributed file system and distributed processing realm. This basic knowledge was tremendously helpful in facilitating the students' understanding of the Hadoop Distributed Files System and the Hadoop distributed processing model.

7.6.7 Family/Career Challenges

This teaching college was founded on the concept of small classes with high student/faculty interactions. Many hours outside normal class were dedicated to tutoring and assisting students master this material. We established a Slack channel that allowed us to communicate during these out of class tutoring sessions. In addition, we dedicated numerous hours outside the class allowing students to engage with us in a one-on-one/small group basis to help them through the challenges presented with this material. Many of the Slack sessions occurred well outside the normal teaching hours when the students could spend time with this material after completing family and career obligations. While for most of the students this was sufficient, we did have a few students who needed additional assistance to master the concepts.

7.7 Prerequisite Effectiveness

Our experimental hypothesis was: Teaching the prerequisite material including Unix, Distributed Systems, remote method invocation, and Operating System basics and then teaching HDFS will improve the student's ability to master this material.

We were concerned that the experimental and control sections might be significantly different and administered a pre-test to both groups. This pre-test shows a 1.7% difference between the two groups. Based on this difference and the shared prerequisites for this course, we concluded that our sections were not statistically different. Running a t-test on the pre-test scores across the groups, we show no significant difference with a t-test of 0.4169.

After teaching the Unix, Hadoop and HDFS concepts to our experimental group and only teaching the HDFS concepts to the control group, we gave the same test as a post-test to these groups. The control group improved their performance by 1.3% while the experimental groups improved 14%. When we examined the t-test for the post-test scores, we had a statistically significant difference of 0.0398. In addition, in the control group 50% of the students scored lower on the post-test than they did on the pre-test. In the experimental group, 100% of the students scored higher on the post-test than they did on the pre-test. This results in a p-value supporting our hypothesis of 0.0005. When calculating the Wilcoxon Signed Rank Test, the Control group has a mean difference of 0.18182 with a p-value of 0.6797. The experimental group shows a mean difference of 2.33333 with a p-value of 0.0313. The p-value for the hypothesis showing improvement in scores in the experimental group on the post-test is 0.0156 while the post-test improvement scores for the control group had a p-value of 0.3398. See Figure 7.9.

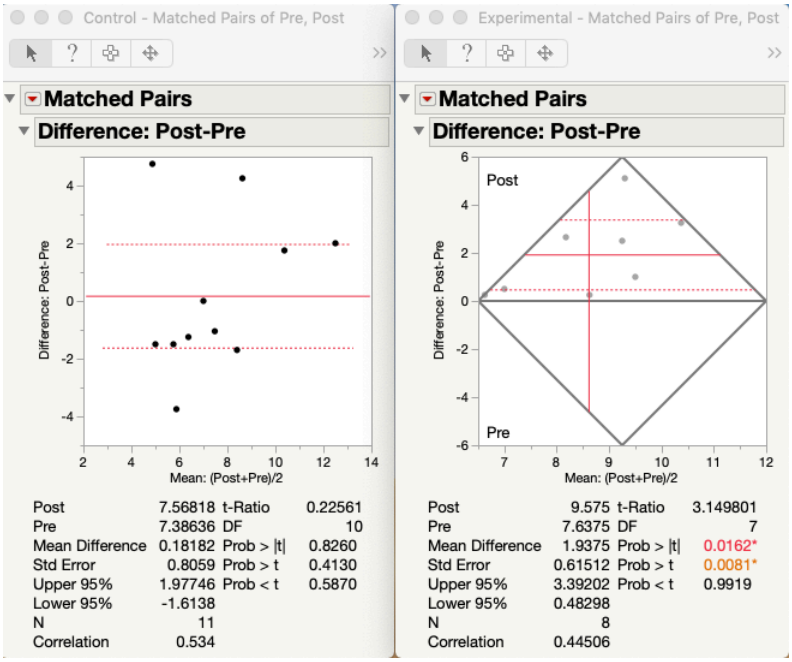

Figure 7.9: Matched Pairs Analysis

Our female students in the experimental section showed an average post-test scores 75% higher than their pre-test score. 100% of the female students in the experimental sections scored higher on the post-test than they did on the pre-test. In the control section, 50% of the female students scored higher and 50% scored lower. The control section female average post-test was 39% higher than the pre-test. A significant finding from the control section is that some of the female students had a decrease in the post-test score of as much as 27%.

The minority students, in our experimental section had an average post-test score 39% higher than the pre-test score. For the control section, there was no change in the average pre-test and post-test scores. In the experimental section, 100% of the minority students improved their scores on the post-test. For the control section, some minority students scores did not change while of those that did change, only 33% of these went up and over 66% of these scores went down.

The final measure of the effectiveness of the proposed prerequisites was the grades on the assigned projects. Each section had one outlier and when these outliers were removed, the experimental section showed a 7% higher grade than the control section.

While our section sizes were not large, we believe the combination of the statistical findings with the information from Chapter 5 on industry perspectives and the daily tasks performed by data scientist validates our hypothesis. While the expectations from industry are widely varied, there is enough supporting evidence from the available industry literature and from the interviews with various industry professionals to confirm the proposed material is viewed as necessary by a large portion of the data science industry.

7.8 Pre/Post Test Specifics

The following section of this dissertation compares the pre/post test scores in the control and experimental sections.

7.8.1 Distributed Systems

On the pre-test, the control section average on the distributed systems portion of the test was 49.62%. The experimental section had a pre-test score of 43.75%. On the post test, the control section had an average score of 46.49% while experimental section had an average score of 68.75%. Differences for control section was a reduction of 3.03% while the experimental section increased by 25%.

7.8.2 Hadoop Distributed File System

For the HDFS portion of our pre/post test questions, we had a surprising result. For the experimental section the average score actually decreased from 75% to 68.75%.

The control section score increased from 59.09% to 66.16%. We will examine this in future experiments to determine the possible reasons for this.

7.8.3 Map/Reduce Paradigm

For the Map/Reduce paradigm, the control section remained constant with pre-test and post-test average score of 60.61%. The experimental section pre-test average score was 60% with an average post-test score of 66.67%.

7.8.4 Data Intensive Basics

The section of the pre-test dealing with Data Intensive computing concepts showed similar results between the control and experimental sections with pre-test average scores of 31% and 22.73% respectively. The remarkable difference was in the post-test average score of 70% for the experimental and 31.82% for the control section.

7.9 Project Description

The final assignment that was given to these students was to take three files and process them. The first file contained seven lines to help them debug their programs. The second file contained 1,674,390 lines. The third file contained 10,001,000 lines. Each line was made up of two three-digit numbers separated by a space. The assignment was to count the number of unique "outlinks" to each of the first number. Obviously with the number of lines, there were multiple entries that mapped to each initial number.

The outcome for this project showed significant understanding of the Hadoop programming process. As shown in Figure 7.10, the students grasped these concepts and were able to successfully complete this assignment. After discussing the project with students, we discovered many of the failures were the result of a lack of study and understanding outside of class. Several of these students had extremely heavy

family/career commitments which resulted in numerous missed classes. Discussing the project with students who did not achieve success revealed that the majority of these failures were because the student did not complete the project due to other commitments and the low weight for the project grade in this course. The majority of these students were in their final semester and enrolled in heavy class loads and chose to simply not complete this project to allow them the time necessary to complete assignments in other classes.

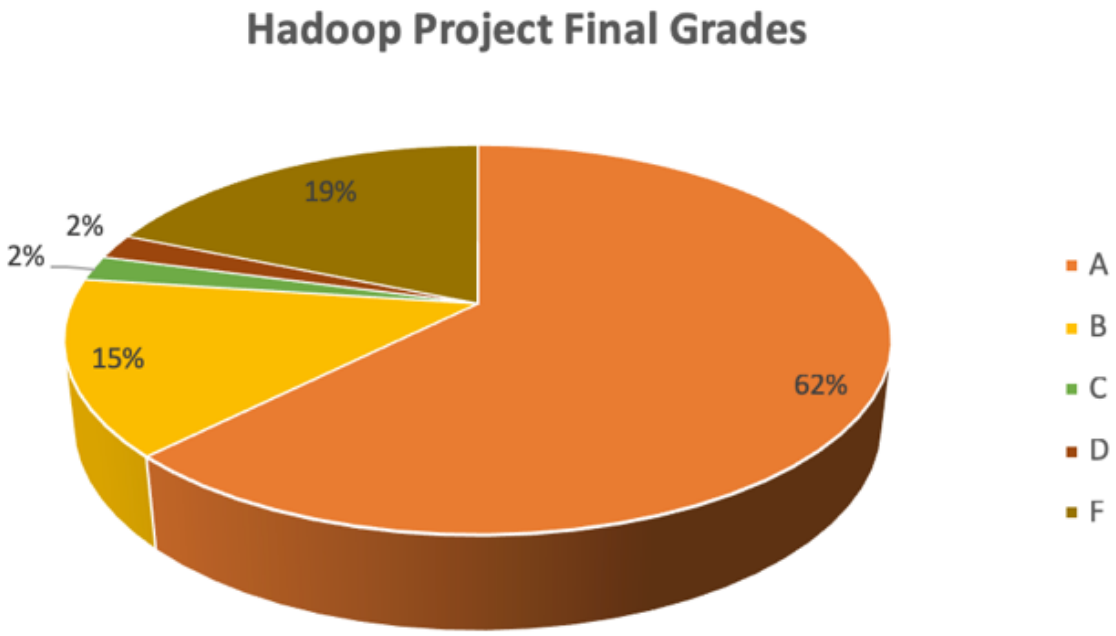

Figure 7.10: Hadoop Final Project Grades

When comparing the project grades for the experimental and control sections, we saw grades in the experimental section were 5% higher. Interestingly, we had two outliers, one in each section. When we remove these outliers, the grades for the experimental section were almost 7% higher.

7.10 Revised Teaching Plan

Based on our findings, we made changes to the teaching plan as detailed in Figure 7.11. We feel this will help students understand basic concepts prior to being introduced to the Hadoop concepts.

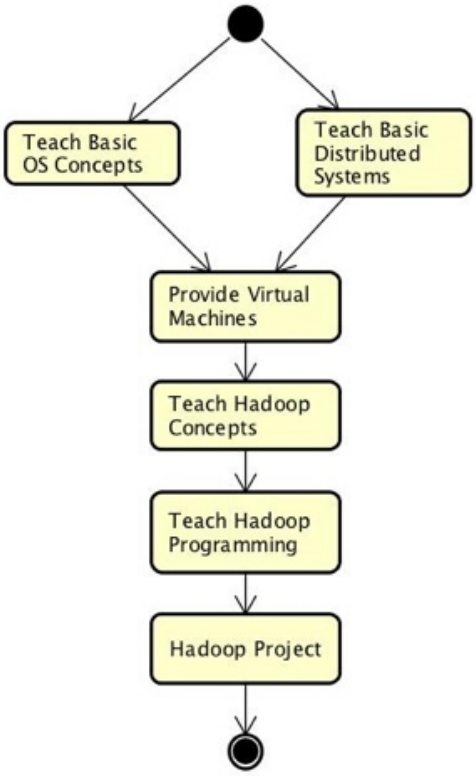

Figure 7.11: Revised Teaching Plan

For future classes, based on the success we had using our experimental approach, we intend to begin by teaching both Operating Systems and Distributed Systems basic concepts.

We plan to provide pre-built environments to students and avoid the difficulties we encountered with setup. We will walk students through the setup and configuration

from a working system rather than starting with a raw system and expect them to complete the entire setup.

Finally, we will add more hands-on examples and class exercises to help students become more familiar with this technology.

7.11 Student Feedback

Students expressed very positive feedback with this project. They felt that this was an important concept to have learned and saw tremendous value for this knowledge in their future careers. One student expressed it very succinctly; “This is a driving force in the industry. The pure volume of information dictates its importance.” Another student expressed their understanding of the importance this way; “Because the future is directed towards big data and an abundant amount of data is created daily, so it is important to learn how to manage this data.”

Students were asked to comment on the different teaching methods employed and we found the in-class walk throughs and the project were the most helpful. “The in-class walk through of an example Hadoop project greatly helped my understanding of the material.” “The project helped the most because it offered hands on experience.”

7.12 Conclusions

Data Science concepts can be taught in an Information Technology program provided core concepts such as Unix and Distributed Systems basics are taught before attempting to teach Data Science fundamentals. Providing pre-built environments was another critical aspect to the success of teaching this material in a course that is already packed with required concepts.

CHAPTER 8

A Foundation for Data Science Education for Information Technology Students

Based on the research covered in the previous chapters, we are proposing the following framework to provide the foundational knowledge for Information Technology students before attempting to teach the Data Science core concepts. This framework includes the following concepts:

- Basic operating systems.
  - Name resolution.
  - Administration.
  - Command line interactions.
  - File system.
- Remote method invocation.
  - CORBA.
  - SOAP.
  - RMI.
- Distributed Systems.
  - Characteristics.
    - Heterogeneity.
    - Transparency.
    - Fault Tolerance.
    - Misconceptions.

- Consistency.
- File System.

- Distributed file systems (Hadoop or other system).
  - Nodes and responsibilities.
  - Failure resilience.
  - Administration.
  - Configuration.
  - Navigation.

RQ1 - Is there a set of material that will improve success in learning data science programming, data collection, and cleaning? Yes, by teaching the material listed above, student understanding is greatly enhanced allowing students to grasp the different data science concepts more easily.

RQ2 - Is there a different set of material needed for minorities and/or women? This research shows that there is no difference in the prerequisite material needed to be provided based on race or gender. The framework described above shows a clear correlation to student success in Data Science related tasks.

RQ3 - Can this be formalized into a framework providing improved student understanding and performance in data science tasks? Yes, the material provided has shown to have a significant impact on student learning when confronted with data science questions and tasks. With the material provided, we had a 100% improvement in understanding of critical material as well as ability to complete programming tasks for data science related problems. Our findings in the classroom were echoed by our industry

experts where they confirmed the need for this foundational material to be present in their knowledge base when arriving in industry.

RQ4 - How can we measure success on data science tasks? Based on pre- and post-test results, we have a clear correlation between student success and the prerequisite material. 100% of the students in the experimental sections improved not only on their post-test scores but also in their ability to perform data science tasks.

RQ5 - How do we address the differences in a traditional CS program and an IT program where many of the core subjects are not required? The prerequisite material presented has shown the knowledge gaps can be bridged by teaching the proposed prerequisite material. This foundational framework has been proven to be effective in teaching Information Technology students data science.

REFERENCES

[1] LinkedIn Communications Team, "August LinkedIn Workforce Report: Data Science Skills are in High Demand Across Industries," 10 August 2018. [Online]. Available: https://news.linkedin.com/2018/8/linkedin-workforce-report-august-2018. [Accessed 8 September 2019].

[2] "Data Scientist," 2019. [Online]. Available: https://www.itcareerfinder.com/it-careers/big-data-scientist.html. [Accessed 1 May 2019].

[3] J. W. Tukey, "The Future of Data Anaylsis," *The Annals of Mathematical Statistics,* vol. 33, no. 1, pp. 1-67, 1962.

[4] D. Donoho, "50 Years of Data Science," *Journal of Computational and Graphical Statistics,* vol. 4, no. 26, pp. 745-766, 2017.

[5] G. Press, "A Very Short History of Big Data," 9 May 2013. [Online]. Available: https://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/#2032eeb265a1. [Accessed 6 March 2018].

[6] P. Naur, Concise Survey of Computer Methods, New York: Petrocelli Books, 1974.

[7] K. D. Foote, "A Brief History of Data Science," 14 December 2016. [Online]. Available: http://www.dataversity.net/brief-history-data-science/. [Accessed 6 March 2018].

[8] The R Foundation, "What is R?," [Online]. Available: https://www.r-project.org/about.html. [Accessed 15 Sep 2019].

[9] M. Cox and D. Ellsworth, "Application-Controlled Demand Paging for Out-of-Core Visualizayion," *VIS '97 Proceedings of the 8th conference on Visualization,* pp. 235-244, 1997.

[10] S. Bryson, D. Kenwright, M. Cox, D. Ellsworth and R. Haimes, "Visually Exploring Gigabyte Data Sets in Real Time," *Communications of the ACM,* vol. 42, no. 8, pp. 83-90, 1999.

[11] G. Press, "A Very Short History of Data Science," 28 May 2013. [Online]. Available: https://www.forbes.com/sites/gilpress/2013/05/28/a-very-short-history-of-data-science/#14db6fc55cfc. [Accessed 6 March 2018].

[12] B. Fry, "Computational Information Design," April 2004. [Online]. Available: http://benfry.com/phd/dissertation-110323c.pdf. [Accessed 11 March 2018].

[13] O. Mendelevitch, C. Stella and D. Eadline, Practical Data Science with Hadoop and Spark, Boston: Addison-Wesley, 2017.

[14] NumFOCUS, "Pandas," 2019. [Online]. Available: https://pandas.pydata.org/. [Accessed 16 Sep 2019].

[15] R. Bryant, R. Katz and E. D. Lazowska, "Big-Data Computing: Creating revolutionary breakthroughs in commerce, science, and society," 2008. [Online]. Available: http://cra.org/ccc/resources/ccc-led-whitepapers/. [Accessed 10 March 2018].

[16] M. E. Driscoll, "THe Three Sexy Skills of Data Geeks," 27 May 2009. [Online]. Available: http://medriscoll.com/post/4740157098/the-three-sexy-skills-of-data-geeks. [Accessed 10 March 2018].

[17] K. Cukier, "Data, data everywhere," *The Economist,* 25 February 2010.

[18] D. Conway, "The Data Science Venn Diagram," 30 September 2010. [Online]. Available: http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram. [Accessed 9 March 2018].

[19] DZone, "Apache Flink vs. Apache Spark," 26 Sep 2017. [Online]. Available: https://dzone.com/articles/apache-flink-vs-apache-spark-brewing-codes. [Accessed 20 Sep 2019].

[20] Apache, "Welcome to Apache HBase," 2019. [Online]. Available: https://hbase.apache.org/. [Accessed 20 Sep 2019].

[21] Apache, "Apache Hive," 2019. [Online]. Available: https://hive.apache.org/. [Accessed 20 Sep 2019].

[22] M. Loukides, What is Data Science?, Sebastopol: O'Reily, 2011.

[23] D. Boyd and K. Crawford, "Critical Questions for Big Data: Provocations for a Cultural, Technological, and Scholarly Phenomenon.," *Information, Communication, & Society,* vol. 15, no. 5, pp. 662-679, 2012.

[24] Apache, "Apache Sqoop," 2019. [Online]. Available: https://sqoop.apache.org/. [Accessed 20 Sep 2019].

[25] Apache, "Apache Flume," 2019. [Online]. Available: https://flume.apache.org/. [Accessed 20 Sep 2019].

[26] J. Chen, Y. Chen, X. Du, C. Li, J. Lu, S. Zhao and X. Zhou, "Big data challenge: A data management perspectve," *Froniters of Computer Science,* vol. 7, no. 2, pp. 157-164, 2013.

[27] P. Guo, "Data Science Workflow: Overview and Challenges," 30 October 2013. [Online]. Available: https://cacm.acm.org/blogs/blog-cacm/169199-data-science-workflow-overview-and-challenges/fulltext. [Accessed 25 November 2017].

[28] G. George, M. R. Haas and A. Pentland, "Big Data and Management From the Editors," *Academy of Management Journal,* vol. 57, no. 2, pp. 321-326, 2014.

[29] L. Cao, "Data Science: Challenges and Directions," *Communications of the ACM,* vol. 60, no. 8, pp. 59-68, 2017.

[30] Task Group on Information Technology Curricula, Information Technology Curricula 2017, IT 2017, New York, NY: ACM, 2017.

[31] ACM, "ACM SIGITE IT Discipline," 8 May 2017. [Online]. Available: http://www.sigite.org/?page_id=22. [Accessed 5 March 2018].

[32] Luther College, "Data Science Major and Minor," Luther College, 21 December 2017. [Online]. Available: https://www.luther.edu/computer-science/data-science-major/. [Accessed 15 March 2018].

[33] Northern Kentucky University, "Data Science B.S," Northern Kentucky University, 2017. [Online]. Available: https://www.nku.edu/academics/informatics/programs/undergraduate/datascience.html. [Accessed 16 March 2018].

[34] Worcester Polytechnic Institute, "Data Science," Worcester Polytechnic Institute, 9 March 2018. [Online]. Available: https://www.wpi.edu/academics/departments/data-science. [Accessed 16 March 2018].

[35] Florida Polytechnic University, "Data Analytics, B.S. Program Description," Florida Polytechnic University, 2017. [Online]. Available: http://floridapolytechnic.catalog.acalog.com/preview_program.php?catoid=11&poid=590. [Accessed 15 March 2018].

[36] R. D. De Veaux, M. Agarwal, M. Averett, B. S. Baumer, A. Bray, T. C. Bressoud, L. Bryant, L. Z. Cheng, A. Francis, R. Gould, A. Y. Kim, M. Kretchmar, Q. Lu, A. Moskol, D. Nolan and R. Pelayo, "Curriculum Guidelines for Undergraduate Programs in Data Science," *Annual Review of Statistics and Its Application,* vol. 4, no. 1, pp. 15-30, 2017.

[37] J. Johnson, L. Tesei, M. Piangerelli, E. Merelli, R. Paci, N. Stojanovic, P. Leito, J. Barbosa and M. Amador, "Big Data: Business, Technology, Education, and Science," *Ubiquity,* vol. 2018, no. July, pp. 2:1-2:13, July 2018.

[38] I. Carmichael and J. Marron, "Data science vs. statistics: two cultures?," *Jpn J Stat Data Sci,* vol. 1, no. 1, pp. 117-138, 1 Jun 2018.

[39] Raymond J. Harbert College of Business, "Business Analytics Degree," Auburn University, 2017. [Online]. Available: http://harbert.auburn.edu/academics/undergraduate/business-analytics/index.php. [Accessed 15 March 2018].

[40] University of Iowa, "Business Analytics and Information Systems Major," University of Iowa, 26 April 2016. [Online]. Available: https://tippie.uiowa.edu/future-undergraduates/majors/business-analytics-information-systems-major. [Accessed 15 March 2018].

[41] Valparaiso University, "Business Analytics," Valparaiso University, 2018. [Online]. Available: https://www.valpo.edu/college-of-business/academics/major-programs/business-analytics/. [Accessed 15 March 2018].

[42] Becker College, "Bachelor of Science in Data Science," Becker College, 29 March 2017. [Online]. Available: https://www.becker.edu/academics/undergrad/school-of-design-technology/data-science. [Accessed March 16 2018].

[43] Southern New Hampshire University, "BS In Data Analytics," Southern New Hampshire University, 2018. [Online]. Available: https://www.snhu.edu/online-degrees/bachelors/bs-in-data-analytics. [Accessed 16 March 2018].

[44] Case Western Reserve, "Data Science," Case Western Reserve, 2018. [Online]. Available: http://case.edu/datascience/. [Accessed 16 March 2018].

[45] University of River Falls Wisconsin, "Data Science and Predictive Analytics," University of River Falls Wisconsin, 2018. [Online]. Available: https://www.uwrf.edu/CBE/Programs/Data-Science-Degree.cfm. [Accessed 17 March 2018].

[46] Massey University, "Bachelor of Science (Data Science)," Massey University, 2018. [Online]. Available: http://www.massey.ac.nz/massey/learning/programme-

course/programme.cfm?prog_id=92411&major_code=PDTSC. [Accessed 17 March 2018].

[47] Arizona State University, "Business Data Analytics," Arizona State University, 9 March 2018. [Online]. Available: https://wpcarey.asu.edu/undergraduate-degrees/business-data-analytics. [Accessed 17 March 2018].

[48] University of San Francisco, "Major in Data Science," University of San Francisco, 14 March 2018. [Online]. Available: https://www.usfca.edu/catalog/undergraduate/arts-sciences/data-science/major. [Accessed 15 March 2018].

[49] Yale College, "Statistics and Data Science," Yale University, 2017. [Online]. Available: http://catalog.yale.edu/ycps/subjects-of-instruction/statistics/. [Accessed 15 March 2018].

[50] Valparaiso University, "B.S. in Data Science," Valparaiso University, 2018. [Online]. Available: https://www.valpo.edu/mathematics-statistics/academics/degree-programs/b-s-in-data-science/. [Accessed 15 March 2018].

[51] Virginia Tech, "Computational Modeling & Data Analytics," Virginia Tech, 2018. [Online]. Available: https://www.ais.science.vt.edu/programs/cmda.html. [Accessed 17 March 2018].

[52] Warwick University, "Undergraduate degrees - Data Science," Warwick University, 2018. [Online]. Available: https://warwick.ac.uk/fac/sci/statistics/courses/datsci. [Accessed 17 March 2018].

[53] Denison, "Data Analytics," Denison, 2018. [Online]. Available: https://denison.edu/academics/data-analytics. [Accessed 16 March 2018].

[54] Ohio State University, "Data Analytics Major," Ohio State University, 2018. [Online]. Available: https://data-analytics.osu.edu/. [Accessed 16 March 2018].

[55] Penn State, "Data Sciences Intercollege Undergraduate Major," Penn State, 2018. [Online]. Available: https://datasciences.psu.edu/. [Accessed 16 March 2018].

[56] College of Charleston, "Data Science Program," College of Charleston, 24 January 2018. [Online]. Available: http://datascience.cofc.edu/. [Accessed 17 March 2018].

[57] I.-Y. Song and Y. Zhu, "Big data and data science: what should we teach?," *Expert Systems 33,* vol. 33, no. 4, pp. 364-373, August 2016.

[58] B. Ramamurthy, "A Practical and Sustainable Model for Learning and Teaching Data Science," in *SIGCSE '16*, Memphis, TN, 2016.

[59] B. Howe, M. Frankin, L. Haas, T. Kraska and J. Ullman, "Data Science Education: We're Missing the Boat, Again," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017.

[60] Business Higher Education Forum, "Data Science and Analytics-Enabled Graduate Competency Map," 9 November 2016. [Online]. Available: http://www.bhef.com/publications/data-science-and-analytics-enabled-graduate-competency-map. [Accessed 19 March 2018].

[61] J. Abdul-Alim, "Leading Educators: All Students Need Data Analytics Course.," 30 March 2017. [Online]. Available: http://diverseeducation.com/article/94506/. [Accessed 09 October 2017].

[62] Business Higher Education Forum, "Investing in America's data science and analytics talent: The case for action," 2017. [Online]. Available: http://www.bhef.com/sites/default/files/bhef_2017_investing_in_dsa.pdf. [Accessed 19 March 2018].

[63] Accreditation Board for Engineering and Technology, "Criteria for Accrediting Computing Programs, 2016-2017," 15 October 2015. [Online]. Available: http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2016-2017/#outcomes. [Accessed 19 March 2018].

[64] University of California, Berkeley, Deparment of Statistics, "Undergraduate Learning Goals," University of California, Berkeley, Deparment of Statistics, 2014. [Online]. Available: https://statistics.berkeley.edu/programs/undergrad/learninggoals. [Accessed 19 March 2018].

[65] Computing Accreditation Commission, "2017-2018 Criteria for Accrediting Computing Programs," ABET, Baltimore, 2016.

[66] L. R. T. S. K. F. Alex Koohang, "Design of an Information Technology Undergraduate Program to Produce IT Versatilists," *Journal of Information Technology Education,* vol. 9, pp. 99 - 113, 2010.

[67] P. Anderson, J. Bowring, R. McCauly, G. Pothering and C. Starr, "An Undergraduate Degree in Data Science: Curriculum and a Decade of Implementation Experience," in *SIGCSE '14*, Atlanta, GA, 2014.

[68] S. Yin and O. Kaynak, "Big Data for Modern Industry: Challenges and Trends," *Proceedings of the IEEE,* vol. 103, no. 2, pp. 143-146, 2 February 2015.

[69] G. Joshi, "Big Data in Insurance," [Online]. Available: https://www.mindtree.com/sites/default/files/mindtree-thought-posts-white-paper-big-data-in-insurance.pdf. [Accessed 7 July 2017].

[70] S. S. Desal and B. Peer, "Big Pharma, Big Data – Big Deal? Yes, Really!," 2015. [Online]. Available: https://www.infosys.com/consulting/insights/documents/big-deal-pharmaceutical-industry.pdf. [Accessed 17 July 2017].

[71] P.-N. Tan, M. Steinbach, A. Karpatne and V. Kumar, Introduction to Data Mining Second Edition, New Your: Pearson, 2019.

[72] M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales and P. Tufano, "Analytics: The real-world use of big data. How innovative organizations are extracting value from uncertain data," October 2012. [Online]. Available: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=GBE03519USEN. [Accessed 6 July 2017].

[73] 451 Advisors, "The Cloud-Based Approach to Achieving Business Value From Big Data," February 2016. [Online]. Available: http://d0.awsstatic.com/analyst-reports/451_Advisors_BigData_White_Paper_2016.pdf?aliId=743069850. [Accessed 7 July 2017].

[74] IBM, "Big data and analytics in travel and transportation," November 2014. [Online]. Available: http://www-07.ibm.com/au/pdf/Travel_Transportation.pdf. [Accessed 7 July 2017].

[75] Infor Manufacturing, "Big Data in manufacturing: A compass for growth," [Online]. Available: http://www.infor.com/content/industry-perspectives/big-data-in-manufacturing.pdf/. [Accessed 7 July 2017].

[76] P. Carnelley and H. Schwenk, "Big Data: Turning Promise Into Reality," October 2016. [Online]. Available: https://www.emc.com/collateral/white-papers/dell-emc-big-data-turning-promise-reality.pdf. [Accessed 17 July 2017].

[77] D. Gutierrez, "Big Data Industry Perspectives for 2017," 21 December 2016. [Online]. Available: insidebigdata.com/2016/12/21/big-data-industry-predictions-2017. [Accessed 7 December 2017].

[78] J. Steunweg- Woords, "Oracle + DataScience.com," 14 May 2018. [Online]. Available: https://www.datascience.com/blog/guide-to-popular-data-science-jobs. [Accessed 12 Sep 2018].

[79] R. Price, L. Ramaswamy and S. Pouriyeh, "Efficient Processing of Range Queries over Distributed Relational Databases," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, Salt Lake, 2018.

[80] D. Karger, E. Lehman, T. Leighton, R. Panigraphy, M. Levine and D. Lewind, "Consistet hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web.," *In Proceedings of the twnty-ninth annual ACM symposium on Theory of computing.,* pp. 654-663, 1997.

[81] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnam, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking,* vol. 11, no. 1, pp. 17-32, 2003.

[82] S. Ratnasamy, J. M. Hellerstein and S. Shenker, "Range Queries over DHTs," *IRB-TR-03-009,* vol. 03, no. 009, pp. 1-2, June 2003.

[83] P. Yalagandula, "Solving Range Queries in a Distributed System," *Tech. Rep. ,* no. TR-04-18, pp. 1-7, 2004.

[84] J. Aspnes and G. Shah, "Skip Graphs," *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms,* vol. 14, no. SODA '03, pp. 384-393, January 2003.

[85] C. Schmidt and M. Parashar, "Flexible Information DIscovery in Decentralized Distributed Systems," *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing,* vol. HPDC.2003, no. 1210032, pp. 226-233, June 2003.

[86] R. Price and L. Ramaswamy, "Challenges and Approaches to Teaching Data Science Technologies in an Information Technology Program with Non-traditional Students," in *2019 IEEE 5th International Conference on Cognitive Machine Intelligence (CogMI)*, Los Angeles, 2019.

[87] I.-Y. a. Y. Z. Song, "Big data and data science: what should we teach?," *Expert Systems 33,* vol. 33, no. 4, pp. 364-373, August 2016.

[88] TREND #5, "Data Science Becomes the Preferred Career Path of the Next Decade," *Trends Magazine,* pp. 29-34, June 2016.

[89] R. Tang and W. Sae-Lim, "Data science programs in U.S. higher education: An exploratory content analysis of program description, curriculum structure and course focus," *Education for Information,* vol. 32, no. 3, pp. 269-290, 01 January 2016.

[90] R. Miller, "If you think big data's big now, just wait," 10 08 2014. [Online]. Available: http://techcrunch.com/2014/08/10/big-data-bound-to-get-really-really-big-with-the-internet-of-things/. [Accessed 25 11 2017].

Appendix A

Institutional Review Board

TO: Mr. Richard Price

FROM: Dr. Andrew Stephenson, IRB Co-Chair

RE: IRB Proposal # 17043

DATE: February 5, 2018

Committee Action: Expedited Review

Study Title: Hadoop Challenges in an Information Technology Program

Your proposal has been reviewed under the committee review process detailed in the policies and procedures of the Institutional Review Board.

I am pleased to inform you that your proposal has been approved. This approval is effective for 12 months. Should your research continue beyond 12 months from this date, you will be required to request and receive continuing approval.

The approved procedure is as follows:

- Someone other than the faculty of record will distribute, and collect informed consent and survey material, and store until final grades are posted.

Renewal: It is the Principal Investigator's responsibility to obtain review and continued approval before the expiration date. Failure to renew your study before the expiration date will result in termination of the study and suspension of related research grants.

Adverse Events: Any serious or unexpected adverse event must be reported to the IRB Chair within 48 hours.

Amendments: Any changes to the protocol, including changes in the research design, equipment, personnel, or funding, must be approved by the IRB committee before they can be initiated.

Personnel: Study personnel must complete training in human subject research. Training can be completed through the NIH or CITI. See the IRB website for details.

http://www.ggc.edu/faculty-and-staff/irb/

You are required to maintain all consent forms in a secure location and to provide a final report of the research to the IRB upon completion of the project.

Appendix B Range-Query Code

MasterLauncher.java

```
import master.RangeQueryMaster;
public class MasterLauncher {
  public static void main(String[] args) throws Exception {
    RangeQueryMaster.main(args);
  }
}
```

SlaveLauncher.java

```
import slave.RangeQuerySlave;
public class SlaveLauncher {
  public static void main(String[] args) {
    RangeQuerySlave.main(args);
  }
}
```

SQLiteException.java

```
package dataLayer;
public class SQLiteException extends Exception {
```

```
        public SQLiteException() {
                super();
        }

        public SQLiteException(String message) {
                super(message);
        }
}
```

SQLiteManager.java

```
package dataLayer;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class SQLiteManager
{
        private Connection conn;
        private Statement stmt;

        public SQLiteManager() throws SQLiteException
        {
```

```
String sDriver = "org.sqlite.JDBC";

try
{
        Class.forName(sDriver);
}
catch (ClassNotFoundException cfne)
{
        throw new SQLiteException("Unable to create the class handling
the database");
}

//Build the URL for SQLite DB
String DB = "words.db";
String jdbc = "jdbc:sqlite";
String dbURL = jdbc + ":" + DB;

//Set db timeout
int timeOut = 30;

try
{
        //Establish a connection to the database
```

```
                conn = DriverManager.getConnection(dbURL);
                //Create a container for the SQL statement
                stmt = conn.createStatement();
                //Set timeout on the statement
                stmt.setQueryTimeout(timeOut);
            }
            catch (SQLException sqe)
            {
                throw new SQLiteException(sqe.getMessage());
            }
    }

    public ResultSet queryDB(String query) throws SQLiteException
    {
        ResultSet rs;
        try
        {
            rs = stmt.executeQuery(query);
        }
        catch (SQLException sqe)
        {
            throw new SQLiteException(sqe.getMessage());
        }
```

```
            return rs;
        }

        public void updateDB(String insert) throws SQLiteException
        {
            try
            {
                stmt.executeUpdate(insert);
            }
            catch (SQLException sqe)
            {
                throw new SQLiteException(sqe.getMessage());
            }
        }
}
```

ReadLargeFile.java

```
package dataPrep;

import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
```

```
import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class ReadLargeFile {

        final static String FILE_NAME = "pg29765.txt";

        final static String OUTPUT_FILE_NAME = "output.txt";

        final static Charset ENCODING = StandardCharsets.UTF_8;

        private static final String SPACE = "\t";

        private String partString = null;

        private String definition = null;

        private Map<String, String> dictKeyWord = new HashMap<String, String>();


        public static void main(String... aArgs) throws IOException{

                ReadLargeFile text = new ReadLargeFile();

                Map<String, String> mapDictionary = new HashMap<String, String>();

                //treat as a large file - use some buffering

                mapDictionary = text.readLargerTextFile(FILE_NAME);

                text.writeLargerTextFile(OUTPUT_FILE_NAME, mapDictionary);

        }
```

```
        private Map<String, String> readLargerTextFile(String aFileName) throws
IOException {
                Path path = Paths.get(aFileName);
                String tempkey = null;
                String tempVal = null;
                Map< String, String> mapKeyDefn = new HashMap<String, String>();
                try (Scanner scanner = new Scanner(path, ENCODING.name())){
                        while (scanner.hasNextLine()){
                                //process each line in some way
                                int countDefn = 0;
                                String temp =scanner.nextLine();
                                if(temp.length()>0){
                                        boolean isKey = checkCapital(temp);
                                        if(isKey){
                                                tempkey = temp;
                                                continue;
                                        }
                                        boolean isDef = checkIfDefinition(temp);
                                        if(isDef){
                                                tempVal = temp;
                                                countDefn =1;
                                        }
                                        if(countDefn==1){
```

```
                                        mapKeyDefn.put(tempkey, tempVal);
                                    }
                                }
                            }
                        }
                        return mapKeyDefn;
                    }

                    private boolean checkIfDefinition(String temp) {
                        if(temp.length()> 10){
                            partString = (String) temp.subSequence(0, 5);
                            System.out.println(partString);
                            if(partString.equalsIgnoreCase("Defn:")){
                                return true;
                            }
                        }
                        return false;
                    }

                    private boolean checkCapital(String temp) {
                        char[] charArray = temp.toCharArray();
                        for(char c : charArray){
                            if(Character.isUpperCase(c))
```

```
                continue;
            else
                return false;
        }
        return true;
    }

    private void writeLargerTextFile(String aFileName,Map<String, String>
mapDict) throws IOException {
        Path path = Paths.get(aFileName);
        try (BufferedWriter writer = Files.newBufferedWriter(path,
ENCODING)){
            for (Map.Entry<String, String> entry : mapDict.entrySet()) {
              String key = entry.getKey();
              String value = entry.getValue();
              writer.write(key+SPACE+value+"\n");
              // ...
            }
        }
    }
    private static void log(Object aMsg){
        System.out.println(String.valueOf(aMsg));
    }
```

```
}
```

RangeQueryMaster.java

```
package master;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.PrintWriter;
import java.util.Vector;
import java.util.Collections;
import java.util.HashSet;
import java.util.InputMismatchException;
import java.util.Iterator;
import java.util.Scanner;

import master.socket.MasterSocket;
import utility.MyTimer;
import utility.RangeBuilder;
import utility.SimpleHash;
import utility.Word;

public class RangeQueryMaster {

        // database servers
```

```
private String[] serverAddr;
private String[] serverName;
public static int numServers = 30;

private MasterSocket[] master;
private static int keyLength = 1;
public static int virtualServer = 100000;
private String[] servers;
private int[] serverMap;
private PrintWriter pw;
private MyTimer rbTimer;
private boolean initialized = false;
private boolean cleared = false;
private boolean connected = false;
private HashSet<String> serversQueried = new HashSet<String>();

public static void main(String[] args) throws Exception {
        RangeQueryMaster rqm = new RangeQueryMaster();
        rqm.runMenu();
}

public void runMenu() throws Exception {
        boolean more = true;
```

```
Scanner input = new Scanner(System.in);
do {
        System.out.println("1. Connect to servers");
        System.out.println("2. Clear the DB");
        System.out.println("3. Load the DB");
        System.out.println("4. Run the test");
        System.out.println("5. Quit");
        int choice = 6;
        try {
                choice = input.nextInt();
        } catch (InputMismatchException ime) {
                choice = 6;
        }

        switch (choice) {
        case 1: {
                connectServers();
                connected = true;
                break;
        }
        case 2: {
                if (connected) {
                        clearDB();
```

```
                    cleared = true;
                } else {
                    System.out.println("You must connect to the
servers before you can initialize the databases");
                }
                break;
            }
            case 3: {
                if (connected) {
                    if (cleared) {
                        initializeTest();
                        initialized = true;
                    } else {
                        System.out.println("You must clear the
databases before you can initialize them.");
                    }
                } else {
                    System.out.println("You must connect to the
servers before you can initialize the databases");
                }
                break;
            }
            case 4: {
```

```
                    if (initialized) {
                        runTest();
                    } else {
                        System.out.println("You must initialize your
database before running tests");
                    }
                    break;
                }
                case 5: {
                    if (initialized) {
                        stopServers();
                    }
                    more = false;
                    break;
                }
                default: {
                    System.out.println("Invalid choice");
                }
                }

            } while (more);
        }
```

```
private void connectServers() {
        // Initialize server array
        buildServers();
        for (int index = 0; index < servers.length; index++) {
                servers[index] = "";
                serverMap[index] = 0;
        }

        // Initialize master socket
        for (int index = 0; index < numServers; index++) {
                master[index] = new MasterSocket(serverAddr[index], 5000);
        }
}

private void initializeTest() throws FileNotFoundException {
        Vector<String> servs = readInitFile();

        String[] temp = servs.get(0).split("=");
        virtualServer = Integer.parseInt(temp[1].trim());
        temp = servs.get(1).split("=");
        keyLength = Integer.parseInt(temp[1].trim());

        rbTimer = new MyTimer();
```

```
            for (int index = 0; index < servers.length; index++) {
                servers[index] = "";
                serverMap[index] = 0;
            }

            // hash servers
            for (int index = 0; index < numServers; index++) {
                int hashCode = hashWord(serverName[index], keyLength);
                servers[hashCode] = serverAddr[index];
            }

            pw = new PrintWriter("report_key_" + keyLength + ".txt");
            pw.println("Starting run. Key length is " + keyLength + ".\n");
            for (int index = 0; index < serverName.length; index++) {
                int hashCode = hashWord(serverName[index], keyLength);
                System.out.println("The hash code for " + serverName[index] + "
is " + hashCode);
                pw.println("The hash code for " + serverName[index] + " is " +
hashCode);
            }
            loadDB();
        }
```

```
private void loadDB() {
        // read file
        ReadFile rf = new ReadFile("output.txt");

        // write data to servers
        Vector<String> words = null;
        try {
                words = rf.readFile();
        } catch (FileNotFoundException fnf) {
                System.err.println("The file \'output.txt\' was not found");
                System.exit(0);
        }

        serversQueried.clear();
        words = parseData(words);
        MyTimer myTime = new MyTimer();
        myTime.startTimer();
        try {
                printWords(words);
        } catch (Exception e) {
                e.printStackTrace();
        }
```

```
                myTime.stop();
                pw.println("\n\nData loading completed in " + myTime.getElapsedTime()
+ " milliseconds\n\n");
                myTime.reset();
        }

        private void runTest() throws Exception {
                MyTimer myTime = new MyTimer();
                // Perform single point queries
                myTime.startTimer();
                Vector<Word> wordList = getWords("REPTILIAN", "REPTILIAN");
                System.out.println("\n");
                pw.println("\n");
                displayWords(wordList);
                myTime.stop();
                pw.println(
                                "\n\nSingle Point query for REPTILIAN completed in " +
myTime.getElapsedTime() + " milliseconds\n\n");
                wordList = new Vector<Word>();
                myTime.reset();
                serversQueried.clear();

                myTime.startTimer();
```

```
                wordList = getWords("CHILD", "CHILD");
                myTime.stop();
                displayWords(wordList);
                pw.println("\n\nSingle Point query for CHILD completed in " +
myTime.getElapsedTime() + " milliseconds\n\n");
                wordList = new Vector<Word>();
                myTime.reset();
                serversQueried.clear();

                myTime.startTimer();
                wordList = getWords("BEACH", "BEACH");
                myTime.stop();
                displayWords(wordList);
                pw.println("\n\nSingle Point query for BEACH completed in " +
myTime.getElapsedTime() + " milliseconds\n\n");
                wordList = new Vector<Word>();
                myTime.reset();
                serversQueried.clear();

                myTime.startTimer();
                wordList = getWords("I", "I");
                myTime.stop();
                displayWords(wordList);
```

```
                pw.println("\n\nSingle Point query for I completed in " +
myTime.getElapsedTime() + " milliseconds\n\n");
                wordList = new Vector<Word>();
                myTime.reset();
                serversQueried.clear();

                // Perform range queries
                myTime.startTimer();
                wordList = getWords("REPTILIAN", "REVENGE");
                myTime.stop();
                displayWords(wordList);
                pw.println("\n\nRange query for REPTILIAN to REVENGE completed in
" + myTime.getElapsedTime()
                                + " milliseconds\n\n");
                wordList = new Vector<Word>();
                myTime.reset();
                serversQueried.clear();

                myTime.startTimer();
                wordList = getWords("APPLE", "APPLICATION");
                myTime.stop();
                displayWords(wordList);
```

```
                pw.println("\n\nRange query for APPLE to APPLICATION completed in
" + myTime.getElapsedTime()
                                + " milliseconds\n\n");
                wordList = new Vector<Word>();
                myTime.reset();
                serversQueried.clear();

                myTime.startTimer();
                wordList = getWords("CAT", "DOG");
                myTime.stop();
                displayWords(wordList);
                pw.println("\n\nRange query for CAT to DOG completed in " +
myTime.getElapsedTime() + " milliseconds\n\n");
                wordList = new Vector<Word>();
                myTime.reset();
                serversQueried.clear();

                myTime.startTimer();
                wordList = getWords("M", "O");
                myTime.stop();
                displayWords(wordList);
                pw.println("\n\nRange query for M to O completed in " +
myTime.getElapsedTime() + " milliseconds\n\n");
```

```
            wordList = new Vector<Word>();
            myTime.reset();

            // exit program
            pw.flush();
            pw.close();

    }

    private void stopServers() {
            for (int index = 0; index < numServers; index++) {
                    master[index].writeLine("Quit");
                    master[index].stopServer();
                    master[index] = null;
            }
    }

    private void buildServers() {
            Vector<String> servs = readInitFile();

            numServers = servs.size() - 2;
            String[] temp = servs.get(0).split("=");
            virtualServer = Integer.parseInt(temp[1].trim());
```

```
            temp = servs.get(1).split("=");
            keyLength = Integer.parseInt(temp[1].trim());

            servers = new String[virtualServer];
            serverMap = new int[virtualServer];
            serverAddr = new String[servs.size() - 2];
            serverName = new String[servs.size() - 2];

            for (int index = 2; index < servs.size(); index++) {
                String[] servAdd = servs.get(index).split("=");
                serverName[index - 2] = servAdd[0].trim();
                serverAddr[index - 2] = servAdd[1].trim();
            }
            master = new MasterSocket[servs.size()];
            System.out.println("Servers built");
    }

    private Vector<String> readInitFile() {
            Vector<String> servs = new Vector<String>();
            try {
                FileReader fr = new FileReader("server.txt");
                Scanner inFile = new Scanner(fr);
                while (inFile.hasNextLine()) {
```

```
                String serverLine = inFile.nextLine();
                if (serverLine.charAt(0) != '#') {
                    servs.add(serverLine);
                }
            }
        } catch (FileNotFoundException fnf) {

        }

        return servs;
    }

    private Vector<String> parseData(Vector<String> words) {
        Vector<String> parseWords = new Vector<String>();

        for (int index = 0; index < words.size(); index++) {
            String[] parsed = parseLine(words.get(index));
            parseWords.add(parsed[0]);
            parseWords.add(parsed[1]);
        }
        return parseWords;
    }
```

```
private String[] parseLine(String line) {
        String[] parsed = line.split("Defn:");
        parsed[0] = parsed[0].trim();
        parsed[1] = parsed[1].trim();
        return parsed;
}

private void printWords(Vector<String> words) throws Exception {
        for (int index = 0; index < numServers; index++) {
                master[index].writeLine("Add words");
        }
        System.out.println("Words size " + words.size() / 2);
        for (int index = 0; index < words.size(); index += 2) {
                String output = words.get(index);
                int hash = hashWord(output, keyLength);
                serverMap[hash]++;
                int serverNum = determineServer(hash);
                int serv = 0;
                do {
                        if (servers[serverNum].equals(serverAddr[serv])) {
                                master[serv].writeLine(output);
                                master[serv].writeLine(words.get(index + 1));
                        }
```

```
                    serv++;
                } while (!servers[serverNum].equals(serverAddr[serv - 1]));
            }

            for (int index = 0; index < numServers; index++) {
                master[index].writeLine("Words complete");
            }
            pw.println("\n");
        }

        private void clearDB() {
            for (int index = 0; index < numServers; index++) {
                master[index].writeLine("Delete words");
                master[index].readLine();
            }
        }

        private Vector<Word> getWords(String startWord, String endWord) throws
Exception {
            Vector<Word> retWords = new Vector<Word>();
            Word tempWord = new Word();
            String sql = "";
            String inline = "";
```

```
// add code to hash words and call correct server
if (startWord.equalsIgnoreCase(endWord)) {
        startWord = buildWord(startWord, true);
        sql = "Select * from words where word = \'" + startWord + "\'";
        int hash = hashWord(startWord, keyLength);
        int serverNum = determineServer(hash);
        int serv = 0;
        do {
                if (servers[serverNum].equals(serverAddr[serv])) {
                        master[serv].writeLine("Get words");
                        master[serv].writeLine(sql);
                }
                serv++;
        } while (!servers[serverNum].equals(serverAddr[serv - 1]));
        while (!inline.equalsIgnoreCase("Words complete")) {
                tempWord = new Word();
                inline = master[serv - 1].readLine();
                tempWord.setWord(inline);
                if (!inline.equalsIgnoreCase("Words complete")) {
                        inline = master[serv - 1].readLine();
                        tempWord.setDefinition(inline);
                        retWords.add(tempWord);
```

```
                }
            }
        } else {
            startWord = buildWord(startWord, true);
            endWord = buildWord(endWord, false);
            RangeBuilder rb = new RangeBuilder(keyLength, this);
            // Vector<String> strs = new Vector<String>();
            rbTimer.startTimer();
            rb.build(startWord, endWord);
            HashSet<Integer> serverList = rb.getServers();
            rbTimer.stop();
            pw.println("Key building for " + startWord + " to " + endWord + "
completed in " + rbTimer.getElapsedTime()
                    + " milliseconds");
            rbTimer.reset();
            // Build the list of servers that need to be queried
            pw.println("\n\nParsing " + startWord + " to " + endWord + " with
a key of " + keyLength + " returned "
                    + serverList.size() + " servers.\n\n");
            Iterator<Integer> serverIter = serverList.iterator();
            while (serverIter.hasNext()) {
                int serv = 0;
                int serverHash = determineServer(serverIter.next());
```

```
                    if (!serversQueried.contains(servers[serverHash])) {

                        while
(!servers[serverHash].equals(serverAddr[serv])) {
                            serv++;
                        }
                        if (servers[serverHash].equals(serverAddr[serv])) {
                            retWords.addAll(readServer(serv,
startWord, endWord));
                        }
                        serversQueried.add(servers[serverHash]);
                    }
                }
            }

            Collections.sort(retWords);
            return retWords;
        }

        private Vector<Word> readServer(Integer serverNum, String startWord, String
endWord) {
            Vector<Word> readWords = new Vector<Word>();
            String inline = "";
```

```
                Word tempWord;
                String sql;
                if (startWord.equalsIgnoreCase(endWord)) {
                        sql = "Select * from words where word like '" + startWord + "%'";
                } else {
                        sql = "Select * from words where word between \'" + startWord +
"!' and '" + endWord + "~\'";
                }
                System.out.println(sql);

                master[serverNum].writeLine("Get words");
                master[serverNum].writeLine(sql);

                while (!inline.equalsIgnoreCase("Words complete")) {
                        tempWord = new Word();
                        inline = master[serverNum].readLine();
                        if (!inline.equalsIgnoreCase("Words complete")) {
                                tempWord.setWord(inline);
                                inline = master[serverNum].readLine();
                                tempWord.setDefinition(inline);
                                readWords.add(tempWord);
                        }
                }
```

```
        return readWords;
    }

    private Vector<Word> readDatabase(String word) throws Exception {
        Vector<Word> readWords = new Vector<Word>();
        String inline = "";
        Word tempWord;
        String sql = "Select * from words where word like \'" + word + "%\'";
        System.out.println(sql);

        int hash = hashWord(word, keyLength);
        int serverNum = determineServer(hash);

        int serv = 0;
        do {
            if (servers[serverNum].equals(serverAddr[serv])) {
                master[serv].writeLine("Get words");
                master[serv].writeLine(sql);
            }
            serv++;
        } while (!servers[serverNum].equals(serverAddr[serv - 1]));

        while (!inline.equalsIgnoreCase("Words complete")) {
```

```
                tempWord = new Word();
                inline = master[serv - 1].readLine();
                tempWord.setWord(inline);
                if (!inline.equalsIgnoreCase("Words complete")) {
                        inline = master[serv - 1].readLine();
                        tempWord.setDefinition(inline);
                        readWords.add(tempWord);
                }
        }
        return readWords;
}

private void displayWords(Vector<Word> words) {
        // for (int index = 0; index < words.size(); index++)
        // {
        // System.out.print(words.get(index).getWord() + " - ");
        // System.out.println(words.get(index).getDefinition());
        //// pw.print(words.get(index).getWord() + " - ");
        //// pw.println(words.get(index).getDefinition());
        // }
        System.out.println("This query returned " + words.size() + " words.");
        pw.println("This query returned " + words.size() + " words.");
}
```

```
public int hashWord(String word, int key) {
        if (word.length() > keyLength) {
                word = word.substring(0, keyLength);
        }

        SimpleHash sh = new SimpleHash();
        return (int) (sh.hashString(word, servers.length));
}

private int hashServer(String word, int key) {
        if (word.length() > keyLength) {
                word = word.substring(0, keyLength);
        }
        int hash = 0;

        for (int index = 0; index < keyLength; index++) {
                hash += word.charAt(index) * 12;
        }

        return hash % servers.length;
}
```

```
        private int determineServer(int hash) {
                while (servers[hash].equals("")) {
                        hash++;
                        if (hash >= virtualServer) {
                                hash = 0;
                        }
                }
                return hash;
        }

        private HashSet<Integer> buildServerList(Vector<String> keys, int keyLen) {
                HashSet<Integer> serverList = new HashSet<Integer>();
                for (int index = 0; index < keys.size() && index < numServers; index++)
{
                        serverList.add(hashWord(keys.get(index), keyLen));
                }

                return serverList;
        }

        private String buildWord(String word, boolean start) {
                char letter = '~';
                if (start) {
```

```
                letter = '!';
            }
            if (word.length() < keyLength) {
                for (int index = 0; index < keyLength - word.length() + 1;
index++) {
                    word += letter;
                }
            } else {
                word = word.substring(0, keyLength);
            }
            return word;
        }
}
```

ReadFile.java

```
package master;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Vector;
import java.util.Scanner;
```

```
public class ReadFile {
	private File inFile;

	public ReadFile(String inputFile) {
		inFile = new File(inputFile);
	}

	public Vector<String> readFile() throws FileNotFoundException {
		Vector<String> dictionary = new Vector<String>();
		FileReader fr = new FileReader(inFile);
		Scanner input = new Scanner(fr);

		while (input.hasNextLine()) {
			dictionary.add(input.nextLine());
		}

		return dictionary;
	}
}
```

MasterSocket.java

```
package master.socket;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;

public class MasterSocket {
        private BufferedReader normalIn;
        public static final int BUFFER_SIZE = 9096;
        private Socket normalSocket = null;
        private String host;
        private int port;
        private PrintWriter normalOut;

        public MasterSocket() {
                host = "127.0.0.1";
                port = 5000;
                startSocket();
        }
```

```
        public MasterSocket(String host, int port) {
                this.host = host;
                this.port = port;
                startSocket();
        }

        private void startSocket() {
                try {
                        System.out.println("Creating socket");
                        normalSocket = new Socket(host, port);
                        System.out.println("Master socket created on " +
normalSocket.getPort());

                        normalOut = new PrintWriter(normalSocket.getOutputStream(),
true);
                        normalIn = new BufferedReader(new
InputStreamReader(normalSocket.getInputStream()));
                } catch (UnknownHostException uhe) {
                        System.out.println("Unknown host specified.");
                } catch (IOException ioe) {
                        System.out.println(ioe.getMessage());
                }
        }
```

```
public void writeLine(String output) {
        normalOut.println(output);
}

public String readLine() {
        String results = "";
        try {
                results = normalIn.readLine();
        } catch (IOException ioe) {
                results = ioe.getMessage();
        }

        return results;
}

public void stopServer() {
        try {
                normalSocket.close();
                normalIn.close();
                normalOut.flush();
                normalOut.close();
        } catch (IOException ioe) {
```

```
                System.out.println("Error closing sockets");
            }
        }
    }
```

RangeQuerySlave.java

```
package slave;

import java.io.IOException;
import java.sql.ResultSet;
import java.sql.SQLException;

import slave.socket.SlaveSocket;
import utility.Word;
import dataLayer.SQLiteException;
import dataLayer.SQLiteManager;

public class RangeQuerySlave {
  private String inLine = "";
  private String inLine2 = "";
  private SQLiteManager sqm = null;
  private SlaveSocket ss = null;
```

```
public static void main(String[] args) {
  RangeQuerySlave rqs = new RangeQuerySlave();
  rqs.ss = new SlaveSocket();
  try {
    rqs.sqm = new SQLiteManager();
  }
  catch (SQLiteException sqe) {
    System.out.println(sqe.getMessage());
    System.exit(0);
  }
  rqs.parseCommand();
}

private void parseCommand() {
  String command = "";
  try {
    ss.acceptConnection();
  }
  catch (IOException ioe) {
    ioe.printStackTrace();
  }
  while (!command.equals("Quit")) {
    try {
```

```
 command = ss.receiveLine();
 System.out.println(command);
}
    catch (IOException ioe) {
 System.out.println("Error receiving commands");
 System.exit(0);
}
if (command.equalsIgnoreCase("Add words")) {
 loadDatabase();
}
if (command.equalsIgnoreCase("Get words")) {
 readDatabase();
 ss.sendLine("Words complete");
}
if (command.equalsIgnoreCase("Delete words")) {
    System.out.println("Deleting words in the DB");
    try {
            clearDB();
            ss.sendLine("Words deleted");
    }
    catch (Exception e) {
            System.out.println(e.getMessage());
    }
```

```
    }
   }
  }

private void clearDB() throws Exception {
        String sql = "Drop table words";
        sqm.updateDB(sql);
        sql = "CREATE TABLE words(word varchar(255) PRIMARY KEY, definition
varchar(900))";
        sqm.updateDB(sql);
}

 private void loadDatabase() {
  try {
   while (!inLine.equalsIgnoreCase("Words complete")) {
    inLine = ss.receiveLine();
    if (!inLine.equalsIgnoreCase("Words complete")) {
     inLine2 = ss.receiveLine();
     System.out.println(inLine + " " + inLine2);
     String sql = "Insert into words(word, definition) values(\""
       + inLine + "\", \"" + inLine2 + "\")";
     sqm.updateDB(sql);
    }
```

```
    }
  }
  catch (IOException ioe) {
    ioe.printStackTrace();
  }
  catch (SQLiteException sqe) {
    sqe.printStackTrace();
  }
}

private void readDatabase() {
  try {
    String sql = ss.receiveLine();
    ResultSet rs = sqm.queryDB(sql);
    while(rs.next()) {
      ss .sendLine(rs.getString("word"));
      ss.sendLine(rs.getString("definition"));
    }
  }
  catch(IOException ioe) {

  }
```

```
    catch (SQLiteException e) {
      e.printStackTrace();
    }
    catch (SQLException e) {
      e.printStackTrace();
    }
  }
}
```

SlaveSocket.java

```
package slave.socket;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class SlaveSocket {
```

```
        private ServerSocket serverSocket;
        private int portNumber;
        private PrintWriter out;
        private BufferedReader in;
        private String defaultDirectory;
        private Socket masterSocket;
        private String inputLine;

        public SlaveSocket() {
                portNumber = 5000;
                masterSocket = null;
                out = null;
                in = null;
                serverSocket = null;
        }

        public void acceptConnection() throws IOException {
                System.out.println("Waiting for a connection");
                serverSocket = new ServerSocket(portNumber);
                masterSocket = serverSocket.accept();
                System.out.println("Connection from master accepted on port " +
masterSocket.getPort());
                out = new PrintWriter(masterSocket.getOutputStream(), true);
```

```
            in = new BufferedReader(new
InputStreamReader(masterSocket.getInputStream()));
        }

        public String receiveLine() throws IOException {
            inputLine = in.readLine();
            return inputLine;
        }

        public void sendLine(String outLine) {
            out.println(outLine);
        }

        public void closeSocket() throws IOException {
            out.close();
            in.close();
            out = null;
            in = null;
            serverSocket.close();
            masterSocket.close();
        }
}
```

MyTimer.java

```
package utility;


public class MyTimer
{
	private long start;
	private long end;

	public MyTimer()
	{
		reset();
	}

	public void reset()
	{
		start = 0;
		end = 0;
	}

	public void startTimer()
	{
		start = System.currentTimeMillis();
	}
```

```
        public void stop()
        {
                end = System.currentTimeMillis();
        }

        public long getElapsedTime()
        {
                return end - start;
        }
}
```

RangeBuilder.java

```
package utility;

import java.util.ArrayList;
import java.util.HashSet;

import master.RangeQueryMaster;

public class RangeBuilder {
        private int keyLength;
        private static int count = 0;
```

```
private static ArrayList<Integer> numValues;

private String origStart, origEnd;

private HashSet<Integer> servers;

private RangeQueryMaster rqm;


public RangeBuilder(int keyLength, RangeQueryMaster rqm) {
        this.rqm = rqm;
        this.keyLength = keyLength;
        numValues = new ArrayList<Integer>();
        // servers = new HashSet<Integer>();
}


public ArrayList<String> build(String start, String end) {
        servers = new HashSet<Integer>();
        ArrayList<String> build = new ArrayList<String>();
        if (start.equals(end)) {
                build.add(start);
                servers.add(rqm.hashWord(start, keyLength));
                return build;
        }
        while (keyLength > start.length()) {
                start += "!";
        }
```

```
                start = start.substring(0, keyLength).toUpperCase();
                while (keyLength > end.length()) {
                        end += "~";
                }
                end = end.substring(0, keyLength).toUpperCase();
                origStart = start;
                origEnd = end;

                return bldString(start, end, 0, build);
        }

        public HashSet<Integer> getServers() {
                return servers;
        }

        private ArrayList<String> bldString(String start, String end, int counter,
ArrayList<String> build) {
                if (servers.size() < RangeQueryMaster.virtualServer) {
                        if (build.size() == 0 || build.get(build.size() - 1).compareTo(end) <
0) {
                                count = counter;
                                int numAdded = 0;
```

```
                if (count == 0) {
                    int index;
                    for (index = start.charAt(0); index <=
end.charAt(0); index++) {
                        numAdded++;
                        build.add("" + (char) index);
                        servers.add(rqm.hashWord("" + (char)
index, keyLength));

                    }
                    numValues.add(numAdded);
                    count++;
                    bldString(start, end, count, build);
                }

                if (count == 1 && count < keyLength) {
                    int first = 0;
                    String last = build.get(first);

                    char stopChar;
                    int outer = 0;
                    int index = 0;
                    numAdded = 0;
```

```
                int firstNum = numValues.get(0);
                for (outer = 0; outer < firstNum; outer++) {
                    if (2 == keyLength) {
                        stopChar = end.charAt(count);
                    } else {
                        stopChar = '~';
                    }

                    char startChar;
                    // code to handle a character before the
second
                    // character in
                    // start when not on the first character
                    if (first == 0) {
                        startChar = start.charAt(count);
                    } else {
                        startChar = ' ';
                    }
                    for (index = startChar; index <= stopChar;
index++) {
                        numAdded++;
                        build.add(last + (char) index);
```

```
                                servers.add(rqm.hashWord(last +
(char) index, keyLength));
                            }
                            first++;
                            last = build.get(first);
                        }
                        numValues.add(numAdded);
                        count++;
                        bldString(start, end, count, build);
                    } else if (count < keyLength && servers.size() <
RangeQueryMaster.virtualServer) {
                        int total = 0;
                        int first = build.size() - numValues.get(count - 1);
                        char stopChar;
                        for (int index = 0; index < numValues.size();
index++) {
                            total += numValues.get(index);
                        }
                        stopChar = end.charAt(count);

                        numValues.add(stopChar - ' ' + 1);
                        for (int outer = 1; outer <= numValues.get(count -
1); outer++) {
```

```
                if (count < keyLength) {
                    stopChar = '~';
                } else {
                    String endInd = build.get(build.size()
- 1);
                    char ender =
endInd.charAt(endInd.length() - 2);
                    char last =
endInd.charAt(endInd.length() - 1);
                    endInd = endInd.substring(0,
endInd.length() - 2) + (ender + 1) + last;
                    for (int inner = 0; inner < count - 1;
inner++) {
                        if (end.charAt(inner) !=
endInd.charAt(inner)) {
                            stopChar = '~';
                        }
                    }
                }
                for (int index = ' '; index <= stopChar;
index++) {
```

```
                                        build.add(build.get(first) + (char)
index);

            servers.add(rqm.hashWord((build.get(first) + ((char) index)), keyLength));
                                            }
                                            first++;
                                        }
                                        numValues.set(count, build.size() - total);
                                        count++;
                                        bldString(start, end, count, build);
                                    }

                                }
                            }
                            return build;
                        }

                        public void setServers(HashSet<Integer> servers) {
                            this.servers = servers;
                        }
                    }
```

SimpleHash.java

```
package utility;


public class SimpleHash {
	public static void main(String[] args) {
		SimpleHash sobj = new SimpleHash();


		// int val = sobj.hash("mnaish",3);
		long val2 = sobj.hashString("manish", 100);
		long val3 = sobj.hashString("mnaish", 100);
		System.out.println(val2);
		System.out.println(val3);
	}


	public long hashString(String string, int M) {
		long hashCode = Math.abs(string.hashCode() * 31);
		return hashCode % M;
	}
}
```

Word.java

```
package utility;
public class Word implements Comparable<Word> {
  private String word;
```

```
private String definition;

public String getWord() {
  return word;
}

public void setWord(String word) {
  this.word = word;
}

public String getDefinition() {
  return definition;
}

public void setDefinition(String definition) {
  this.definition = definition;
}

@Override
public int compareTo(Word wrd) {
  int comparison = word.compareTo(wrd.getWord());
  return comparison;
}
```

}