Maynooth University
National University of Ireland Maynooth

Endless Data

A dissertation submitted for the degree of
Doctor of Philosophy

By:

Jonathan Dunne

Under the supervision of:

Dr David Malone

Hamilton Institute
Maynooth University
Ollscoil na hÉireann, Má Nuad

October 6, 2018

Yet is it far better to light the candle than to curse the darkness.

- William Lonsdale Watkinson.

Contents

List of Tables

Declaration

I hereby declare that I have produced this manuscript without the prohibited assistance of any third parties and without making use of aids other than those specified.

The thesis work was conducted from January 1, 2015 to October 6, 2018 under the supervision of Dr. David Malone in Hamilton Institute, National University of Ireland Maynooth. This work was funded and supported by employer IBM.

Maynooth, Ireland,

October 6, 2018

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Dr. David Malone for the continuous support of my PhD studies and related research, for his patience, motivation, use of grammar, and immense knowledge. His guidance helped me immensely research and writing of this thesis. I could not have imagined having a better advisor and mentor.

My sincere thanks also goes to Prof. Ken Duffy and Dr. Oliver Mason, for their support, guidance and for the many interesting discussions over lunch.

I would also like to thank all of my colleagues (especially Ashling, Hazel Karl), from the Hamilton Institute, it was a privilege to work with so many wonderful people on a daily basis. I would especially like to thank Rosemary Hunt and Kate Moriarty for all of their help, guidance and for helping me to cope with the administrative requirements of a PhD.

From the bottom of my heart I must also thank my wife Helen and my children Ethan and Ceola for their support throughout all of my crazy endeavours. Without their support, this thesis would not be possible. I'd also like to thank my Mum and Dad their belief in me was unwavering.

Finally, I must also thank all of my friends who supported me through this work, it wasn't easy on them, so I am especially grateful for your patience, understanding and keeping me sane throughout.

Abstract

Small and Medium Enterprises (SMEs), as well as micro teams, face an uphill task when delivering software to the Cloud. While rapid release methods such as Continuous Delivery can speed up the delivery cycle: software quality, application uptime and information management remain key concerns. This work looks at four aspects of software delivery: crowdsourced testing, Cloud outage modelling, collaborative chat discourse modelling, and collaborative chat discourse segmentation. For each aspect, we consider business related questions around how to improve software quality and gain more significant insights into collaborative data while respecting the rapid release paradigm.

Glossary

Term	Meaning
Agile development	Agile development describes a set of principles for software development whereby requirements evolve through collaboration with cross-functional teams. Agile encourages an iterative development style. Agile promotes the idea that early delivery can be achieved at regular intervals throughout the product development lifecycle. First examples of iterative development can be traced back to the late 1950s however it was not until the idea of rapid software delivery gained wider adoption [1]. See chapter 2 for more details.
Business Support System (BSS)	Herbert Simon states that the BSS is the "connection point" between external relations (customers, partners and suppliers) and an enterprise's products and services [2].
Continuous delivery	Continuous Delivery (CD) is the ability to get changes of all types – including new features, configuration changes, bug fixes and experiments – into production, or into the hands of users, safely and quickly in a sustainable way [3].

DevOps	Is a practice that highlights the collaboration between software development and infrastructure personnel. DevOps may also refer to a team which has a core function to build, deploy and maintain a Cloud infrastructure. These definitions are referenced from some sources such as "What is DevOps?" by Mike Loukides [4] and "The Phoenix Project" by Kim et al. [5].
Downtime (Outage)	The Alliance for Telecommunications Industry Solutions (ATIS) define the term downtime as periods when a system is unavailable. Downtime or outage duration refers to a period that a system fails to provide or perform its primary function [6].
Field defects	Refers to all defects found by customers or internal test teams using a software product post-release [7].
Functional Testing	Testing which is focused on the specified functional requirements and does not verify the interactions of system functions [7].
Maintenance window	In information technology and systems management, a maintenance window is a period designated in advance by the technical staff, during which preventive maintenance that could disrupt service may be performed. This definition is also provided by ATIS [6].
Micro team	Deshpande defined a Micro team as one that typically consists of three or four persons, with just a single or at most two developers, a business analyst and a project manager or surrogate customer [8].
Performance Testing	In software engineering, Performance testing is performed to determine how a system performs regarding responsiveness and stability under a particular workload [7].
Queuing theory	Is the study of events that form waiting lines or queues. In queuing theory, a model is constructed so that queue lengths, inter-arrival and service times can be predicted [9, 10, 11].

SME	Enterprise Ireland define a small enterprise as an enterprise that has fewer than 50 employees and has either an annual turnover and an annual balance sheet total not exceeding €10m. A medium enterprise is an enterprise that has between 50 employees and 249 employees and has either an annual turnover not exceeding €50m or an annual balance sheet total not exceeding €43m [12].
Startup	Croll and Yoskovitz in their book lean analytics state that a startup is an organisation formed to search for a repeatable and scalable business model [13].
System Testing	Testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System test, unlike Functional testing, validates end-to-end system operations within the broader environmental context. Therefore system testing should be conducted on an environment, which closely mimics's customer behaviour[7].
Tiger Team	The term tiger team was first introduced in 1964. Dempsey et al. defined a tiger team as a team of undomesticated and uninhibited technical specialists, selected for their experience, energy, and imagination. Tiger teams are assigned to relentlessly track down every possible source of failure in a spacecraft subsystem [14].
Waterfall model	A waterfall model is a stepped process used in software development. The design process works through some phases (cascading like a waterfall) as follows: conception, initiation, analysis, design, construction, testing, production/implementation and maintenance. The term was initially coined by Herbert D. Benington in 1956 [15]. The first formal use of Waterfall was by Winston Royce in 1970 [16].

CHAPTER 1

Introduction

In this chapter, we discuss the motivations behind the work of this thesis and provide an overview of the material presented in the following chapters.

1.1 Motivation

Before the advent of the Cloud and rapid software release models, software was developed and delivered to customers via a method called Waterfall. Waterfall is a non-iterative design process used in software development. One of the disadvantages of Waterfall was the time taken from development, test and delivery to the customer (i.e. a release every 12 to 36 months).

With the broad adoption of consumer computing in the 1990s, customer demand precipitated a move away from Waterfall to a leaner "little and often" approach. Agile development introduced the concept that a team should have a feature or component ready for release at periodic stages during its development cycle. The adoption of agile development practices allowed for incremental releases to made at a much faster rate.

A refinement of agile came about with the round-the-clock availability of Cloud computing and services in the late 1990's / early 2000's. Cloud computing adopted the practice of using a network of remote servers to store, manage, and process data, rather than a local server or a personal computer. With the "always available" paradigm, new development practices were required. A

new development model known as continuous delivery (CD) was created. CD differs from agile development in that development teams develop software that is ready for release at any time during development.

Delivering software for the Cloud represents a challenge for both micro teams, Small Medium Enterprises (SMEs) and startups, in part due to the rapid release methods (i.e. CD) adopted and the numerous ways in which software defects can be detected.

Likewise, as applications are hosted on a Cloud-based infrastructure, production outages (critical software defects) can occur in a variety of ways due to the complex nature of distributed computing.

Instant messaging is a popular form of real-time communication between groups. Adoption of such collaboration tools took off during the mid-1990s with tools such as AOL Instant Messenger, ICQ, PowWow and IRC. As businesses realised the potential of real-time communication, additional corporate offerings were released: IBM Sametime, MSN Messenger and Yahoo!.

The core attraction for businesses was chiefly the ability for teams regardless of size or location to collaborate on a wide range of topics (e.g. Cloud outage remediation). However, with the growth of next-generation solutions such as Slack, IBM Workspace and Microsoft teams, large volumes of text data is generated. Making sense of this data can be a challenge to teams, given the lack of inbuilt analytical tooling.

We extend previous work by studying field defects, production outage events and real-time collaboration data. Through empirical research using multiple enterprise and open-source datasets, we provide a series of frameworks that can be used to turn endless streams of data into high-value information.

1.2 Overview

This thesis is divided into the following chapters:

Chapter 1: Introduction

This is the introductory chapter and broad discussion of the context that, in part motivates this proposed work.

Chapter 2: Literature Review

For the literature review chapter, background and related research are reviewed and discussed.

Chapter 3: Social Testing

This chapter presents two studies of software testing conducted as part of the release of a Cloud-based enterprise application. The concept of social testing is introduced, and we demonstrate how this idea can be used to reduce field defects. We pay particular attention as to why this type of study is important for small teams.

Chapter 4: Outage Modelling

This chapter discloses details of a study conducted into Cloud outages, using an enterprise dataset. Both outage inter-arrival and service times are modelled. We extend the idea of critical defect discovery to Cloud outages. These events have the potential to disrupt a Cloud service for a temporary or extended period of time.

Chapter 5: Outage Simulation In this chapter, we take the inter-arrival and service times from our Cloud outage study and simulate the arrival of future outages events. From this simulation staffing requirements to manage such events can be inferred.

Chapter 6: Chat Discourse Modelling

This chapter presents a framework that can be used to model real-time chat discourse using both parametric and non-parametric methods. Collaboration applications are used to diagnose Cloud outage events. Modelling conversation can serve as an analogue to Cloud outage service times.

Chapter 7: Chat Discourse Segmentation and Boundary Identification

This chapter describes a novel technique to segment chat conversations to provide an improved degree of understanding for topic modelling. We also consider a text classification framework to identify chat conversation boundaries. Text mining of chat discourse can be useful to understand key terms in prior Cloud outage engagements. Identification of conversation boundaries

may provide more targeted topic modelling on a per discussion basis.

Chapter 8: Conclusions

This last chapter summarises the key findings, learning outcomes and pathways for future work.

1.3 Research Questions

Over the course of this thesis, we want to use the eight case studies to answer the following research questions within the context of a CD/CI rapid software delivery model.

Chapter 3: Social Testing

Our first research question is centred around understanding what types of field defects do customers find once the software has been released into the field? Our second research question asks how many field defects are located within the first four weeks of a release to the field?

Chapter 4: Outage Modelling

Chapter four focuses on modelling the inter-arrival and service time of Cloud outage events. Our third research question asks what type of probability distribution can be used to model the inter-arrival time between Cloud outages? While our fourth research question asks what kind of probability distribution can be used to model the service time to resolve Cloud outages?

Chapter 5: Outage Simulation

With an inter-arrival and service time result obtained from chapter 4, can we use this result to simulate a simple queue model to identify queue busy times and if so, can this simulation provide a comparable degree of precision to that of observed outage data? This is our fifth research question.

Chapter 6: Chat Discourse Modelling

Chapter 6 explores the use of real-time chat software by DevOps teams to diagnose and resolve Cloud outage events. We ask, by modelling conversation durations of real-time chat discourse, do these conversation durations serve as an analogue to the service times modelled in chapter 4?

Chapter 7: Chat Discourse Segmentation and Boundary Identification

Exploring chat discourse further, chapter 7 poses two final research questions. Our seventh question asks whether by segmenting chat discourse using burst and reflection periods, can more words be made available to topic model software? Additionally, if more words are made available, does a higher number of words mean a higher level of understanding to both a human reader and a computer? Finally, our eight research question asks, whether we can use a supervised machine learning algorithm to detect the boundaries (i.e. start and end) and if so to what degree of precision?

1.4 Publications

Over the lifetime of this research, the following conference papers, journal (both peer-reviewed) and poster session have been presented. Work is presented by publication type for ease of reference.

Journals

- "Obscured by the cloud: A resource allocation framework to model cloud outage events." *Journal of Systems and Software* 131 (2017): 218-229.

Conference Papers

- Social testing: A framework to support adoption of continuous delivery by Small Medium Enterprises. (CSCESM 2015: The Second International Conference on Computer Science, Computer Engineering, & Social Media)
- Are you being served: A Framework to manage Cloud outage repair times for Small Medium Enterprises. (27th Irish Signals and Systems Conference - 2016)
- Social dogfood: A framework to minimise Cloud field defects through crowdsourced testing. (28th Irish Signals and Systems Conference - 2017)

- Different every time: A framework to model real-time instant message conversations. (21st Finnish-Russian University Cooperation in Telecommunications conference - 2017)
- Bundles: A framework to optimise topic analysis in real-time chat discourse (24th International Conference on Collaboration and Technology - 2018)
- Hello and Goodbye: A framework to identify conversation boundaries in real-time chat discourse (29th Irish Signals and Systems Conference - 2018)

Poster Sessions

- Of queues and cures: A solution to modelling the inter time arrivals of cloud outage events (36th Conference on Applied Statistics in Ireland - 2016)

CHAPTER 2

Literature review

In this chapter, a comprehensive review of background related literature in the field of crowdsourced testing, Cloud outages, topic modelling and chat message segmentation is presented. Important statistical concepts and methodologies are also discussed.

2.1 Introduction

With any body of research it is important to outline and recognise a) the prior related work upon which this research is based and b) the key statistical methods that were used to derive results. This chapter is intended to ground the reader with a background in the areas of: continuous software delivery, bug bounty programs, crowdsourced testing, Cloud outage events and real-time chat discourse. Additionally, this review chapter will cover key statistical techniques that were required to produce both analysis and results upon this research is based (e.g. Distribution fitting, Heavy-tailed analysis, Queuing theory). Finally the review will provide discussion on prior research conducted in the field of software reliability, outage detection and chat discourse segmentation.

2.2 Cloud Computing

The following section provides some background information on two common Cloud services: SaaS and PaaS. We then review high profile Cloud outages

that have made the media headlines in recent times. Finally, this section concludes with an in-depth look at relevant studies in the field of repairable systems modelling, queuing theory and Cloud outage studies.

2.2.1 Software as a Service (SaaS)

SaaS is defined as a delivery and licensing model in which software is used on a subscription basis (e.g. monthly, quarterly or yearly) and where applications or services are hosted centrally [17].

The key benefits for software vendors are the ability for software to be available on a continuous basis (on-demand) and for a single deployment pattern to be used. It is this single deployment pattern that can greatly reduce code validation times in pre-release testing, due to the homogeneous architecture. Central hosting also allows for rapid release of new features and updates through automated delivery processes [18].

SaaS is now ubiquitous, while initially adopted by the large software vendors (e.g. Amazon, Microsoft, IBM, Google and Salesforce) many SMEs are now using the Cloud as their delivery platform of choice [19].

2.2.2 Platform as a Service (PaaS)

PaaS is defined as a delivery and platform management model. This model allows customers to develop and maintain Cloud-based software and services without the need for building and managing a complex Cloud-based infrastructure.

The main attraction of PaaS is that it allows micro teams and SMEs to rapidly develop and deliver Cloud-based software and services. While focusing on their core products and services micro teams and SMEs are less distracted by having to design, build and service a large complex Cloud-based infrastructure.

However, one drawback of PaaS is that a micro team or SME may not have a full view of the wider infrastructure. Therefore if an outage event occurs at an infrastructure level (e.g. Network, Loadbalancer) a micro team or SME may be unaware of the problem until the issue is reported by a customer.

Many companies now offer PaaS as their core service. Once seen as the preserve of a large organisation (e.g. Amazon EC2, Google Apps and IBM

Bluemix) a number of smaller dedicated companies also offer PaaS (e.g. Dokku, OpenShift and Kubernetes) [20].

2.2.3 Cloud outages

A Cloud outage is the amount of time that a service is unavailable to the customer. While the benefits of Cloud systems are well known, a key disadvantage is that when a Cloud environment becomes unavailable, it can take a significant amount of time to diagnose and resolve the problem. During this time the platform may be unavailable for some or all customers.

One of the first Cloud outages to make the headlines in recent times was the Amazon outage in April 2011. In summary, the Amazon Cloud experienced an outage that lasted 47 hours, the root cause of the issue was a configuration change made as part of a network upgrade. While this issue would be damaging enough for Amazon alone, some consumers of Amazon's Cloud platform (Reddit, Foresquare) were also affected. [21]

Dropbox experienced two widespread outages during 2013 [22, 23]. The first in January, users were unable to connect to the service. It took Dropbox 15 hours to restore a full service. No official explanation as to the nature of the outage was given. The second occurred in May; users were unable to connect to the service. This outage lasted a mere 90 minutes. No official explanation was provided.

While improvements have been made in the area of redundancy, disaster recovery and ring-fencing of key critical services, the big players in Cloud computing are not immune to outages. As of mid-2016, a number of high profile outages were catalogued by the CRN website. [24] [25] Table 2.1 provides a summary.

2.2.4 Studies related to Cloud outages

A number of studies have been conducted on Cloud outages and the time observed to resolve problems in repairable systems in recent times. We review the key papers in this field and discuss their core contributions.

Yuan et al. [26] performed a comprehensive study of distributed system failures. Their study found that almost all failures could be reproduced on reduced node architecture and that performing tests on error handling code

Table 2.1: Summary of high profile Cloud outages during 2015 & 2016

Company	**Outage Time**	**Outage Details**
Amazon	10 hours	Local storms in Australia caused Amazon Web Services to lose power. This resulted in some EC2 instances to fail, which affected both SaaS and PaaS customers.
Apple iCloud	12 hours	A Domain Name Server (DNS) error meant that users were unable to make purchases.
Apple iCloud	7 hours	iCloud unavailable / poor performance affected 200 million users.
Microsoft	Several days	Users reported issues accessing their Cloud-based mail services. The defect was identified, and a software fix was applied. This fix proved unsuccessful, after that a secondary fix was developed and applied which was successful.
Twitter	8 hours	Users experienced general operational problems after an internal software update was applied to the production system with faulty code. It took Twitter 8 hours to debug and remediate the defective code.
Salesforce	10 hours	European Salesforce users had their services disrupted due to a storage problem in their EU Data Centre. After the storage issue was resolved, users reported performance degradation.
Starbucks	Unspecified	Scheduled maintenance resulted in the tilling system going off-line.
Symantec	24 Hours	A portal to allow customers to manage their Cloud security services became unavailable. The exact nature of the outage was undisclosed. Symantec was required to restore and configure a database to bring the system back online.
Verizon	40 hours	Scheduled maintenance to improve overall reliability.
Windows Azure	2 hours	A network infrastructure outage resulted in loss of service for all central US users.

could have prevented the majority of failures. They conclude by discussing the efficacy of their own static code checker as a way to check error-handling routines.

Hagen et al. [27] conducted a study into the root cause of an Amazon Cloud outage on April 21st, 2011. Their study concluded that a configuration change was made to route traffic from one router to another, while a network upgrade was conducted. The backup router did not have sufficient capacity to handle the required load. They developed a verification technique to detect change conflicts and safety constraints, within a network infrastructure before execution.

Li et al. [28] conducted a systematic survey of public Cloud outage events. Their findings generated a framework, which classified outage root causes. Of the 78 outage events surveyed they found that the most common causes of outages included: System issues, i.e. (human error, contention) and power outages being the primary root cause of failure.

Sedaghat et al. [29] modelled correlated failures caused by both network and power failures. As part of the study, the authors developed a reliability model and an approximation technique for assessing a services reliability in the presence of correlated failures.

Potharaju and Navendu [30] conducted a similar study concerning network outages, with focus on categorising intra and inter-data centre network failures. Two key findings include: Network redundancy is most effective at the inter-datacentre level, and interface errors, hardware failures and unexpected reboots dominate root cause determination.

Bodik et al. [31] analysed the network communication of a large-scale web application. Then proposed a framework that achieves a high fault tolerance with reduced bandwidth usage in outage conditions.

Carcary et al. [32] conducted a study into Cloud computing adoption by Irish SMEs. The key findings of the study were as follows: Almost half the 95 SMEs surveyed had not migrated their services to the Cloud. Of those SMEs that had migrated they had not assessed their readiness to adopt Cloud computing. Finally, the study noted that the main constraints for SMEs adoption of Cloud computing were: Security/compliance concerns, lack of IT skills and data

protection concerns.

2.2.5 Studies related repair time modelling and system reliability

The purpose of repair time modelling, is to understand how long it takes to repair a fault in a repairable software system. By studying prior repair time events, we can model and predict the time taken to repair future events. We now discuss six of the most relevant contributions to the domain of repair time modelling and system reliability.

Synder et al. [33] conducted a study on the reliability of Cloud-based systems. The authors developed an algorithm based on a non-sequential Monte Carlo Simulation to evaluate the reliability of large-scale Cloud systems. The authors found that by intelligently allocating the correct types of virtual machine instances, overall Cloud reliability can be maintained with a high degree of precision.

Kenney [34] proposes a model to estimate the arrival of field defects based on the number of software defects found during in-house testing. The model is based on the Weibull distribution which arises from the assumption that field usage of commercial software increases as a power function of time. If we think of Cloud outages as a form of field defect, there is much to consider in this model. For example, the arrival of Cloud outages in the field could be modelled with a power law distribution (e.g. Pareto distribution) as a starting point.

Kleyner and O'Connor [35] propose an important book regarding reliability engineering. While the emphasis is placed on measuring reliability for both mechanical and electrical/electronic systems, the authors do broaden their scope to discuss reliability of computer software. One aspect of interest is their discussion of the lognormal distribution and its application in modelling for system reliability with wear out characteristics and for modelling the repair times of maintained systems.

Almog [36] analysed repair data from twenty maintainable electronic systems to validate whether either the lognormal or exponential distribution would be a suitable candidate distribution to model repair times. His results showed

that in 67% of datasets the lognormal distribution was a suitable fit, while the exponential was unsuitable in 62% all of datasets.

Adedigba [37] analysed the service times from a help desk call centre. Her study showed that the exponential distribution did not provide a reasonable fit for call centre service times. However, a log-normal distribution was a reasonable fit for overall service times. Her study also showed that a phase-type distribution with three phases provided a reasonable fit for service times for specific jobs within the call centre job queue.

Alsoghayer and Djemame [38] propose a mathematical model to predict the risk of failures with Grid and Cloud-based infrastructures. The model uses the observed mean time to failure and the mean to repair for prediction. The authors found the best model to predict the time between failures is a Weibull distribution, while the repair (service) times as best modelled by a lognormal distribution.

2.3 Software Development Models

The software development process typically involves dividing software development work into distinct phases (i.e. Requirement gathering, design, implementation and verification) in order to deliver software to the end user. This framework and methodology began during the late 1960's [39]. In this section, we review three methodology's that have been used in recent times: Waterfall, Agile and Continuous Delivery. While our focus is on the latter method, the following section should serve as a potted history of software development models.

2.3.1 Waterfall

The Waterfall development model is a stepped approach to software development. As a development phase is completed, the process moves down (like a waterfall) to the next phase or step. A Waterfall method will typically include the following phases: Requirements gathering, design, implementation, testing, integration, deployment and maintenance. The term was initially coined by Herbert D. Benington in 1956 [15]. The first formal use of Waterfall was by Winston Royce in 1970 [16].

Waterfall is seen as an inflexible development model, due mainly to the idea that phases cannot be started until a preceding phase has been completed. This can lead to extended periods of development and test before release [40]. In some cases, a software project could enter an 18 month or longer development duration before market release [41]. Waterfall is seen as a traditional development model and was used extensively during the 1980's and 1990's.

2.3.2 Agile

Agile software development is a set of values and methodologies whereby software solutions are developed using an iterative based approach by utilising self-organising cross-functional teams. Agile development was popularised by the publication "Manifesto for Agile software development" [42] in 2001. Agile is comprised of two main frameworks: Kanban and Scrum.

Kanban is a method to visualise workflows within teams. Kanban typically focuses on lanes of work. These lanes are collections of tasks that have either been completed, are in progress or to be completed. By using this visual methodology, teams can graphically represent the progress of an overall project. Kanban can be used to determine overall project effort and productivity at each stage [43].

Scrum proposes that teams are divided into squads containing three to nine developers. These developers break their actions into discrete tasks that can be completed in a fixed timeframe called a "sprint". Daily stand-up meetings are used to assess progress with each task. Tasks that cannot be completed within the sprint timeframe can be moved to a backlogged state and is referred to as technical debt [44].

As with multiple frameworks, there are advocates on both sides for using Kanban or Scrum. However, teams are most successful when they adopt elements of both frameworks [45].

In some ways, Agile is seen as the antithesis of Waterfall, because a specific feature component can be made available from design, development, test and delivery in a relatively short timeframe (e.g. two or three week sprint). It should be noted that this iterative approach of design to release is mostly an embodiment of design thinking [46].

2.3.3 Continuous Delivery

CD is an approach to software development that allows software companies to develop, test and release software, in short, discrete delivery cycles. Releasing software with a low number of changes allows the rapid validation and release of a software product. CD Employs two methodologies; continuous test automation (CA) — the practice of employing an automated test script to validate delivered code and continuous integration (CI) — the practice of merging developer streams into a consolidated mainline, which allows software to be developed and tested to a high standard (due to the low level of code churn), and facilitates a swift release cycle. CD is used as part of a new wave of development, test, deployment and release strategies for Cloud-based software services. Key evangelists for CD include Facebook, Google and Netflix [47].

2.4 Crowdsourced Testing & Field Defect Studies

In this section, we review some themes related to the area of both software testing and studies related to field defect detection. We concentrate our literature review, relative to testing methods (i.e. Crowdsourced testing, Bug Bounties and Dogfood testing) that we believe will enhance and improve the test experience for SMEs and micro teams. It is not an exhaustive review of software testing as a whole. To explore software testing in greater detail is beyond the scope of this thesis. Finally, we round off this section with a review of studies related mainly to review of field defects.

2.4.1 Crowdsourced testing

Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call [48].

Nebling et al. [49] present a study of Crowdstudy, a toolkit for crowdsourced testing of web pages. By crowdsourcing numerous individuals, data was collected on how users habits differed when engaging with a website.

Vukovic [50] conducted a study of crowdsourcing services for the Cloud. While Amazon's Mechanical Turk and Innocentive appear to have the most sup-

ported features, most of the frameworks fall short in facilitating the dynamic formation of globally distributed teams.

Liu et al. [51] conducted a study into the use of crowdsourcing for usability testing compared to traditional face-to-face methods. Their study found the quality of results from crowdsourcing was not as good as those face-to-face testing. However, crowdsourcing still represents value for design/development teams with limited time and money.

Zogaj et al. [52] present a case study with a crowdsourcing company who specialise in outsourcing software testing to specific groups. Their research found that there were three key challenges: managing the process, managing the crowd and managing the technology. By using an intermediary to manage all aspects of the process from procurement of individuals, monitoring of test progress to addressing technology skill gaps ensured a smooth end to end process.

2.4.2 Bug Bounty Programs

A bug bounty program is a scheme whereby software companies offer a reward to users that find defects within their software. The benefit to software companies is that it incentivises users to find defects (typically security vulnerabilities) before they are exploited by the general user base [53]. The *bugcrowd* website contains a list of current bug bounties offered by software companies. As of March 2018, 344 companies are listed as having some form of reward and gift system for user found vulnerabilities [54]. Bug bounty schemes are not limited to start-up companies or open source projects. Some high profile software companies which participate in bug bounty schemes include Facebook, Google and Microsoft.

There have been some well-known bug bounty programmes. For example, Donald Knuth a computer scientist and creator of the TeX computer system [55] devised a bug bounty program (Knuth reward checks) where the reward doubled every year in value to a maximum of $327.68 in the form of a cashier's check. A second well-known bounty is related to D.J. Bernstein who is a cryptologist and programmer of qmail [56]. In 1997 he offered $500 to the first individual who could publish details of security exploits within his latest release of qmail. To date, no one has found any vulnerability.

2.4.3 Eating your own dogfood

Eating your own dogfood is a term given to the internal usage of a software product before release to the customer. The idea is that regular internal usage will improve overall software quality.

Warren Harrison [57] the then editor in chief of IEEE Software, mentions that Microsoft was one of the first companies to aggressively adopt the practice of "Eating your own dogfood" when developing their Windows platform in the early 1990's. Harrison also discusses the pros and cons of adopting a dog food approach to internal testing.

Adam Moskowitz [58] discussed the idea of dog fooding in the magazine ";login:". In the realm of system administration, Moskowitz provides some practical examples of how increased internal testing can help improve shell scripting and tooling.

Schmidt and Varian [59], in a Newsweek article, outlined ten rules, which they believe will drive success within Google over the next quarter of a century. They attribute the success of Gmail to the fact that it was extensively tested by the majority of Google employees over a several month period.

Prlić and Procter [60] in a Public Library of Science computer biology journal, also outline ten rules from the open development of scientific software. Rule three mentions how software in development should be used as an end product and not merely to demonstrate a solution. In other words, the software should be consumable by customers with a broad range of backgrounds rather than a specific cohort.

Jackson and Winn [61] researched the field of research data management platforms. In building a large complex platform with many API endpoints, they cite internal usage of the in-development platform coupled with the adoption of Agile practices such as 'Continuous Integration'[62] as key methods in defect detection.

2.4.4 Studies related to defect detection

We now review seven studies related to customer reported defects. Interestingly, at the time of our literature review, none of the software studied was developed using a CD release model.

Brooks and Robinson [63] performed a study on customer reported GUI defects found on two industrial software systems. Their work focused on the impact, location and resolution times of customer defects. Their study compared these factors from both in-house and customer defects. They found that in-house testers and customers found the same types of defects. 60% of the total defects found were in the GUI while the remaining 40% were application defects. Finally, that customers had to wait 170 days on average, for defects to be fixed.

Moritz [64] conducted a study of customer defects raised within a large industrial telecommunications software component. Her work focused on the analysis of customer defects found over a 10-year period with a goal of understanding how to improve future test quality. She reviewed whether defect regression, test phase, new functionality, load testing and environment were factors in a customer defect being raised. Her study found first, that the in-house system test environments and test use cases did not accurately match customer configurations or usage conditions. Second, that regression testing was inadequate; tests plans typically focused on new features, which left test exposures within legacy components. Finally, existing test methods were not suitable for finding customer defects.

Gittens et al. [65] studied the efficiency of in-house software testing by investigating the scope and coverage of system and regression testing. They examined some factors, such as the number of defects found in-house, by the customer and code coverage. Firstly that in-house test coverage should cover between 71 – 80% of the product code base prior to shipping. Secondly, that in-house tests coverage does not always overlap with customer usage areas. Thus there is a gap between in-house and customer product usage. Finally, that greater in-house test coverage does not automatically translate into fewer customer defects found. The authors demonstrated that test coverage needs to be specifically targeted to reduce field defects.

Musa [66] developed a technique for Software Reliability Engineered Testing (SRET), which was implemented on the *Fone Follower* project at AT&T. Musa used a SRET method to classify defects found into four levels of severity based on their impact on the end user. Defect severity rates from prior regression testing were then used to guide future test coverage.

Sullivan and Chillarege [67] compared the types of customer defects found in Database Systems (DBS) and Operating Systems (OS). Their study looked at some factors including; error type, trigger and defect type. They had some key findings. Firstly they found that legacy DBS and OS had a similar number of high severity defects. Secondly, that newer DBS had a higher rate of high severity defects.

Adams [68] conducted a study of customer defects from nine products over a five-year period. He found that customer defects were typically discovered shortly after the product was released. He surmised that these defects would have taken many person months to find had they been tested on a single machine. He concluded that these customer defects would have been very difficult to find using existing test methods.

Riungu et al. [69] performed research into the challenges Cloud computing presents to software testing. One concern raised was the human effort to test software with 24/7 availability. Automation aside, they mentioned the need for some level of manual testing to be conducted round the clock—an idea not easily implemented by SMEs.

2.5 Data modelling

Over the years a vast array of techniques that have to used to model many forms of data. In this section, we shall discuss two specific approaches. The first is a parametric technique whereby we assume that our data belongs to a given distribution type. A second approach is a non-parametric approach, whereby there is no underlying assumption about our data. Sometimes such data is said to be distribution free. Also discussed in this section are methods to test the goodness of fit of a parametric approach, the background behind heavy-tailed data, and finally a technique to assist in the fitting of count data where the data may be under or over-dispersed.

2.5.1 Distribution Fitting

Probability distribution fitting is the procedure of selecting a statistical distribution that best fits a data set generated by a random process. For example, if we have random data generated from a process (e.g. the time to service

a Cloud outage event) and we want to know what distribution can be best describe our data, distribution fitting can be a useful exercise.

In statistics, we can estimate the parameters of a statistical model using prior observations. For the purposes of providing background on estimation, and distribution fitting, two methods are discussed briefly.

The first method is called the method of moments. This method uses expected values of a random variable (a moment) from a population. A sample is then taken from the population and subsequent moment is estimated. The sample moments are used to make estimates about an unknown population. This idea was first proposed by Pafnuty Chebyshev in 1887 [70].

The second method is called Maximum likelihood estimation (MLE). MLE is a method to estimate the parameter values of a model by determining the parameter values that maximise the likelihood. This technique was first proposed by Ronald Fisher in the 1920's [71], with a subsequent formal proof by Samuel Wilks in 1938 [72].

2.5.2 Goodness of Fit Testing

If a suitable probability distribution can be found to fit a data set, of interest is how well the distribution fits that data. Some methods have been developed to assess the goodness of fit of a distribution to a data set. We shall discuss three of the main tests briefly.

The Cramér–von Mises criterion [73][74] is a non-parametric test which examines the goodness of fit of a cumulative distribution function (CDF) compared to that of an empirical distribution function (EDF). Using a significance test, we can test a hypothesis of whether a data set is drawn from a given probability distribution

The Kolmogorov–Smirnov [75] test quantifies a distance between the EDF of the sample and the CDF of the reference distribution, or between the EDF of two samples. The idea being that the closer the distance between the two, the better the fit.

The Anderson–Darling [76][77] test is a statistical test of whether a given sample of data is drawn from a given probability distribution. This test is a

modification of the Kolmogorov–Smirnov test as it gives more weight to the tails of data.

2.5.3 Heavy Tailed Estimation

In probability theory, heavy-tailed distributions are distributions whose tails are not exponentially bounded. In fact, these distributions often have much heavier tails, for example, a Pareto or generalised extreme value distribution. For such distributions a tail index, which is essentially the shape parameter of a distribution is used to make inferences about the underlying data.

Hill [78] proposes one of the first methods to infer tail behaviour of a distribution function. This work is valuable in that no prior assumption of the type of distribution is required before inference. His tail estimation technique is one of the standard methods for measuring the index of a heavy-tailed distribution.

Pickands [79] provides a method to make inferences about the tail of a probability distribution function. This technique is applied to all continuous distribution functions. Pickands method is an alternative method to calculate the index of a heavy-tailed distribution.

Nair et al. [80] discuss the idea that heavy-tailed data and their corresponding distributions are a more common occurrence. They also discuss various techniques to model distributions from heavy-tailed datasets.

2.5.4 Hurdle Distribution

Hurdle distributions are a class of distributions for count data that can help manage datasets with a large number of zeros or a count dataset that exhibits either over-dispersion or under-dispersion. Mullahy [81] proposes the idea of a hurdle model which provides a more natural means to model over or under-dispersed count data.

2.5.5 Kernel Density Estimation

For datasets which do not fit a known distribution family, a non-parametric approach can be taken. One such approach is Kernel Density Estimation (KDE). In KDE, a range of kernel (weighting) functions are applied to a dataset plotted as a histogram. The kernel functions are divided into various

widths (bandwidth). The goal is to choose the most appropriate kernel bandwidth and function shape that best fits the data. Both Rosenblatt [82] and Parzen [83] are credited with creating KDE in it's current form. Some significant contributions have been made in the field of KDE. These are discussed briefly below.

Kernel performance is measured by either the mean integrated squared error (MISE) or the asymptotic mean integrated squared error (AMISE). Epanechnikov [84] proposed a parabolically shaped kernel that minimises AMISE and is therefore optimal. Kernel efficiency is now measured in comparison to the Epanechnikov kernel.

Silverman [85] proposes an improved method for bandwidth selection. In his study, if a Gaussian basis function is used to approximate univariate data, and if the underlying density is Gaussian, the optimal choice for the bandwidth parameter is the standard deviation of the samples. This method is known and Silverman's rule of thumb or the Gaussian approximation.

Sheather and Jones [86] provided an improved method for data-based selection of the bandwidth in KDE. Their paper included a new bias term in their bandwidth estimate, which provides improved performance for a broad set of cases.

2.5.6 Linear Regression

Simple linear regression [87] is a statistical method that allows us to summarise and study relationships between two continuous variables. One variable, denoted x, is regarded as the predictor or independent variable. The other variable, denoted y, is regarded as the response or dependent variable. The formula for simple linear regression is provided below:

$$Y_i = \alpha + \beta x_i + \epsilon_i \tag{2.1}$$

α is the intercept parameter, β is the slope parameter. Both α and β are collectively known as the *regression coefficients* of the regression line. Finally ε is a random error component. One of the assumptions for regression analysis is that the residuals are normally distributed (i.e. ϵ_i $N(0, \sigma^2)$).

The process of simple linear regression may begin with plotting both sets of variables (i.e. x, y) in the form of a scatter plot. This method is used to obtain a visual representation of both variables. Additionally, we can use a scatterplot graph to determine a positive relationship (when the variable x increases, so does y) or a negative relationship (as variable x increases, y decreases). Finally, where more than one predictor or independent variable is used the process is known as multiple linear regression.

A Poisson distribution is used to model non-negative integers, and can be used to model the distribution of data in the form of counts. It tends to be the first distribution considered for count data. It follows that if we have a need to predict response data in the form of counts. Poisson regression can be used to explore such data relationships. We use the same general form of x, as the independent variable and y as the response variable. However the core difference is that we employ the use of a log link function. The formula for Poisson regression is provided below (Note: $\vartheta(x) = \alpha + \beta x$).

$$log(E(Y_i)) = \vartheta(x_i) (2.2)$$

2.5.7 Studies related to modelling real-time communication messaging

The purpose of modelling real-time message communication is to understand the structure of how messages are sent, received and composed. For example, by modelling the inter-arrival of messages can help us know if there is an underlying property to such data. By using either parametric and non-parametric methods, we determine if a given dataset can be drawn from a known distribution to provide fine-grained analysis. In this section, we discuss seven studies related to modelling of real-time message data.

Dewes et al. [88] conducted a study to better understand network traffic dynamics by examining Internet chat systems. While their primary research output was to demonstrate how to separate chat traffic from other Internet traffic, the authors conducted an analysis of the inter-arrival times of chat messages. The authors' hypothesis was as follows: Are the inter-arrival times of chat messages consistent with an exponential distribution? The hypothesis

was rejected due to lack of evidence. However, they found the inter-arrival times were more consistent with a heavy-tailed distribution.

Lukasik et al. [89] modelled time series data of tweets to understand if a reliable prediction model could be derived to predict future tweets. Their research found that by modelling tweet inter-arrival times under a log-Gaussian Cox process, a higher degree of predictive precision could be achieved. The authors also found that mining text from tweet messages can improve inter-arrival time prediction.

Vande Kerckhove et al. [90] provided research into the field of inter-arrival times of electronic communication. The authors investigated the level of inter-event dependence between postings and whether a Markovian process would be suitable to model the memory effect observed in inter-arrival online activities. For their study, the authors used social media data from Twitter and Reddit. Their research concluded that by allowing dependence between message wait times provides for more precise modelling than by fitting against a power-law distribution alone.

Markovitch and Krieger [91] compared the non-parametric estimation of the probability density function of long-tailed distributions from Internet-based traffic against existing parametric methods. The authors found that neither a Pareto nor an exponential model was a suitable fit to their underlying data. Additionally by using both a Parzen–Rosenblatt kernel and a histogram of variable width (a polygram) a more suitable fit was achieved.

Maioroda and Markovitch [92] discuss the non-parametric estimation of a heavy-tailed probability density function by a variable bandwidth kernel estimator. The authors discuss two approaches: A preliminary transformation to provide an information estimation of tail density and a discrepancy method based on the Kolmogorov-Smirnov statistic to evaluate the bandwidth of the kernel estimator. The authors use Internet-based traffic to validate their models.

Wang [93] presents a how-to article on visualising the inter-arrival times of tweets. Using the R programming language the author describes the process to collect, visualise and determine if the inter-arrival times can be modelled by a Poisson process.

Burnap et al. [94] consider the models to predict information flow size survival using data derived from the popular social networking site Twitter. To model predict flow size and survival rates, zero-truncated negative binomial and Cox regression models were used. This study did not model the distribution of tweet data. However, it is noted that the number of tweets studied and their survival duration were both heavy-tailed.

2.6 Machine Learning

Machine learning is an area of computer science that allows computers to learn the outcome of a task without being explicitly programmed to do so [95]. Tom Mitchell describes machine learning as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [96]. Machine learning begins by observing data directly and using this knowledge to infer patterns in data and make decisions or predictions on additional examples. The phrase "Machine Learning" was first coined by Arthur Samuel in 1959 while working at IBM [97].

Machine learning can employ various algorithms to provide output. These algorithms can be categorised as supervised, unsupervised, semi-supervised and reinforcement.

Supervised learning is probably the most common type of algorithm used today. The core idea that prior labelled data (known as training data) is used to generate a corresponding matching output from another set of data (known as test data). Ideally, the machine learning algorithm can generalise the training data in a meaningful way to determine a classified label from unseen data [98]. Examples of supervised algorithms include naïve Bayes, Support vector machine (SVM), decision trees and random forest.

With unsupervised learning, prior data is neither labelled or classified. In this case, an algorithm is used to cluster data around data that is inferred as being similar. The main aim of unsupervised learning is to provide exploratory data analysis by inferring hidden patterns or groups [99]. Examples of unsupervised algorithms are k-means clustering, Gaussian mixture models and hidden Markov models.

Semi-supervised learning is a hybrid of both algorithms, typically a small amount of labelled data is used to cluster data into a set of known groups. Reinforcement learning is an approach whereby an algorithm interacts with an environment to determine a set of actions to maximise a reward. This method allows for a level of 'ideal behaviour' to be inferred [100].

Finally, statistical learning theory is a framework for machine learning building on the fields of statistics and functional analysis [101]. Statistical learning theory is concerned with the problem of finding a predictive function based on data. Statistical learning theory has led to successful applications in fields such as object and speech recognition [102].

In the following subsections, we discuss four popular supervised learning algorithms in more detail.

2.6.1 Naïve Bayes

A Naïve Bayes classifier is a type of machine learning algorithm that uses Bayes theorem. This algorithm makes a strong (naïve) assumption of independence between each pair of features (i.e. an individual measurable property of a phenomenon being observed) [103]. Despite the seemingly over-simplified assumptions of independence, naïve Bayes has shown to be useful with solving real word problems most notably in the field of document classification and email spam filtering [104].

2.6.2 Decision Trees

In machine learning a decision tree (DT) is a non-parametric supervised learning algorithm that uses observed data to make decisions about unseen data [105]. As the name implies, labelled data is represented in the form of a tree. Decisions are made by the underlying data in the form of a branch (feature intersections) and leaf-like (class labels) structure. Some advantages of DT's over other types of classifiers include: As DT's (and their results) is visualised graphically, therefore they are easy to interpret. Furthermore, DT's use a white box model, as a result it is easier to modify the algorithm parameters and determine an improvement or degradation in classification performance. DT's do have problems in that the models created do not generalise to variations not seen in a training set [106].

2.6.3 Support Vector Machine

Support vector machine (SVM) is an algorithm used in machine learning to solve classification and regression problems [107]. SVM represents labelled observations as points in space. As the points are plotted the algorithm determines what line best separates the labelled classes. This separation point is also known as a hyperplane. Ideally, a hyperplane with the largest distance between both sets of classes is preferable, as this makes it easier to distinguish between classes.

If a classification problem presents whereby a line is unable to separate the labelled classes successfully, SVM can use a non-linear classification. A *kernel trick* is used to perform a data transformation to create a high dimension feature space. As a result the hyperplane may be extended to a curve or a series of curves. The kernel trick was initially proposed as far back as 1964 by Mark Aizerman [108]. Vapnik et al. are credited with successfully incorporating the kernel trick to SVM in the early 1990's [109].

Due to SVM's flexibility, the algorithm has been used to solve many real world problems in the field of text and image classification. For example, in both the fields of face recognition [110] and bioinformatics [111], SVM has been shown to be an effective method to classify both facial features and proteins in distinct studies.

2.6.4 Random Forest

Random forest is an ensemble algorithm classifier that is used in machine learning. Rather than being a distinct classifier in of itself, it is a collection of techniques used for classification [112]. Random forests work by building a number of decision trees and outputs a class that is the most prevalent. Random forests are used to correct the behaviour of decision trees, that overfit to their training data [101]. An additional ensemble method includes a technique known as bagging. Bagging involves a random selection of features at the training stage [113]. This random selection is used to reduce the level of variance in an estimator.

2.7 Natural Language Processing

Natural Language Processing (NLP) is a field of research that explores how computers can be used to understand and interpret natural language text or speech [114]. If a computer has a good understanding of how humans write and talk, we can use software to classify, summarise and topic model text. To help a computer understand text syntax more efficiently, a number of techniques are employed. We discuss four of the most common methods next.

Tokenisation is a process of converting a sequence of characters (e.g. message discourse) into a series of tokens (strings with an assigned meaning) [115]. Therefore, before any analysis is conducted on a text corpus, the text is divided into linguistic elements such as words, punctuation, numbers and alpha-numerics [116].

Stop words are words which are filtered out before or after processing of text discourse [117]. Stop words typically refer to the most common words in a language; there is no consensus or master list of agreed stop words. The website "ranks.nl" provides lists of stop words in forty languages [118]. Hans Luhn, one of the pioneers in the field of information retrieval, is credited with creating the concept of stop words [119].

Stemming is a method of collapsing inflected words to their base or root form [120]. For example, the words: fishing, fished and fisher, could be reduced to their root fish. The benefit of stemming can be seen as follows: If one is interested in term frequency, it may be easiest to merely count the occurrences of the word fish rather than its non-stemmed counterparts. One drawback of stemming is that inflections contain information. In the field of information retrieval, it may be sufficient to understand whether the root of a word (i.e. fish) is mentioned. In the field of text classification, a more precise result may be achieved by using the inflected word, rather than it's root alone. We shall explore these pros and cons in more detail in chapter 7.

Lemmatisation is the process of grouping together the inflected words, for analysis as a single entity [121]. On the surface this process may look like the opposite of stemming; however, the main difference is that stemming is unaware of the context of the words and thus, cannot differentiate between words that have other meanings depending on context. For example, the

word “worse” has “bad” as its lemma. This link is missed by stemming as a dictionary lookup is needed. Whereas, the word “talk” is the root of “talking”. This reference is matched in both stemming and lemmatisation. We note the the level of pre-processing required is guided by the problem domain (i.e. Information retrieval or document classification)

2.7.1 Corpus Linguistics

Corpus linguistics is the study of language as expressed in corpora (i.e. collections) of “actual use” text. The core idea is that analysis of expression is best conducted within its natural usage. By collecting samples of writing, researchers can understand how individuals converse with each other. One of the most influential studies in this field was conducted by Kučera and Francis [122]. The authors analysed an American English Corpus, that involved analysis techniques from linguistics, psychology and statistics.

2.7.2 Topic Modelling Tools

Latent Semantic Analysis (LSA) is a method that allows for a low-dimension representation of documents and words. By constructing a document-term matrix, and using matrix algebra, one can infer document similarity (product of row vectors) and word similarity (product of column vectors). The idea was first proposed by Landauer et al. in 1998[123].

In 1999 Hofman proposed a statistical technique of two-mode and co-occurrence data [124]. In essence, his Probabilistic Latent Semantic Analysis model (PLSA), allowed a higher degree of precision for information retrieval than standard LSA models. This is due to the introduction of a novel Tempered Expectation Maximisation technique that used a probabilistic method rather than matrices for fitting. However, one drawback of the PLSA method, is that, as the number words and documents increase, so does the level of overfitting.

Latent Dirichlet allocation (LDA) is a generative statistical model that allows topics within a text corpus to be represented as a collection of terms [125]. At its core, LDA is a three-level hierarchal Bayesian model, in which each item in an array is modelled as a finite mixture over an underlying set of topics. Blei et al. first proposed the idea in 2003.

2.7.3 Studies Related to Topic Mining of Small Text Corpora

In Chapter 7, we present a case study of a corpus based approach of topic modelling small text messages. In this present section, we review seven notable studies carried out in the field of information retrieval and NLP. Five of the seven studies involve short or small text sources.

Jivani conducts a comparative study of eleven stemmers, to compare their advantages and limitations [126]. The study found that there is a lot of similarity regarding performance between the various stemming algorithms. Additionally, a rule-based approach may provide the correct output for all cases, as the stems generated may not always be accurate words. For linguistic stemmers their output is highly dependent on the lexicon used, and words outside of the lexicon are not stemmed correctly.

Naveed et al. [127] investigates the problem of document sparsity in topic mining in the realm of micro-blogs. Their study found that ignoring length normalisation improves retrieval results. By introducing an "interestingness" (level of re-tweets) quality measurement also improves retrieval performance.

The Biterm topic model is explicitly designed for small text corpora such as instant messages and tweet discourse [128]. Conventional topic models such as LDA implicitly capture the document-level word co-occurrence patterns to reveal topics, and thus suffer from the severe data sparsity in short documents. With these problems identified, Yan et al., proposed a topic model that a) explicitly models word co-occurrence patterns and b) uses the aggregated patterns in the whole corpus for learning topics to solve the problem of sparse word co-occurrence patterns at document-level.

Yin et al. [129] discuss the problem of topic modelling short text corpora such as tweets and social media messages. The core challenges are due to sparse, high-dimensional and large volume characteristics. The authors proposed a Gibbs Sampling algorithm for the Dirichlet model (GSDMM). The authors demonstrated that a sparsity model could achieve better performance than either K-means clustering or a Dirichlet Process Mixture Model for Document Clustering with Feature Partition.

Sridhar [130] presents an unsupervised topic model for short texts using a

Gaussian mixture model. His model uses a vector space model that overcomes the issue of word sparsity. The author demonstrates the efficacy of this model compared to LDA using tweet message data.

Topic Modelling of Short Texts: A Pseudo-Document View by Zuo et al. [131] propose a probabilistic model called Pseudo-document-based topic model (PTM) for short text topic modelling. PTM introduces the idea of a pseudo-document to implicitly aggregate short texts against data sparsity. By modelling these pseudo-documents rather than short texts, a higher degree of performance is achieved. An additional sparsity enhancement is proposed that removes undesirable correlations between pseudo-documents and latent topics.

Schofield and Mimmo [132] investigate the effects of stemmers on topic models. Their research concluded that stemming does not help in controlling the size of vocabulary for topic modelling algorithms like LDA, and may reduce the predictive likelihood. The authors suggest that post-stemming may exploit nuances specific to corpora and computationally more efficient due to the smaller list of words for input.

2.7.4 Text Classification

Text classification is a subset of document classification, whereby text is required to be labelled as a specific class or category. Classes are selected from an established hierarchy of existing classes. For example, text may be classified by subject, author or emotive tone.

The classification task was traditionally a manual one. However, in recent times due to the advent of large corpora of text data and relatively cheap computing power, the task is mainly conducted using a machine learning algorithm with varying degrees of success [133].

Today text classification by computers is used to solve many concrete problems such as sentiment detection (i.e. detecting positive or negative film reviews), email sorting (i.e. sort emails sent by family, business colleagues or a spambot).

2.7.5 Studies Related to Text Segmentation

The purpose of text segmentation is to identify specific regions of text within a corpus. The benefit of such a practice is to aid in the field of information retrieval, where topic boundary identification is a crucial problem. We

discuss some of the leading contributions to the domain of topic boundary identification briefly.

One of the first studies (1991) in the field of text segmentation was conducted by Morris and Hirst [134]. The authors focused on the problem of lexical cohesion (chains of related words), by using a thesaurus as a knowledge base for computing lexical chains. Additional early contributors in the field of lexical cohesion include Kozima [135], who proposed a lexical cohesion profile, and Reynar [136] who outlined an improved method of locating discourse boundaries based on the previous method of lexical cohesion and a graphical technique call dotplotting.

Some years later, additional techniques have been used to tackle the problem of partitioning text into coherent segments. Beeferman et al. [137] introduced an exponential model to extract features that are correlated to the presence of boundaries. Their study used *Wall Street Journal* news articles and television news story transcripts. Galley et al. [138] propose a discourse segmentation technique using for multi-party conversations. Their lexical cohesion algorithm demonstrated reasonable results when text extracted from the Brown corpus [1].

In more recent times (2012 onwards), new researchers used different techniques to research text segmentation. Nguyen et al. [139] proposed a Bayesian non-parametric model to discover the topics used in a conversation, topic shift and a person specific tendency to introduce new topics. The authors used transcripts from the 2008 presidential debate and a television programme called Crossfile. Brooks et al. [140] used a machine learning approach to identify effective state (e.g. joy excitement, confusion, frustration, anger and annoyance) on chat logs, using comprised of discussion from an astrophysics institute. Schmidt and Stone [141] use a combination of techniques (i.e. Latent semantic analysis, text tiling, and pause detection) to detect topic changes in Internet Relay Chat (IRC) chat logs, with limited success.

Rounding off our studies in this section, Uthus and Aha [142] surveyed research on the analysis of multi-participant chat. The authors conclude that chat data is difficult to analyse due to its unique characteristics due to the many

[1] http://clu.uni.no/icame/brown/bcm.html

problems the medium presents (e.g. Chat room feature processing, thread disentanglement, topic detection, summarisation and user profiling). This has caused many traditional text analysis techniques to prove unsuccessful. The authors suggest that given its prevalence of social communication, the domain represents an exciting research topic.

2.8 Conclusion

It may be argued, with the seemingly endless amount of data generated today, the challenge to make sense of this data is insurmountable. It is fair to acknowledge that data generation of many kinds is on the increase. Nevertheless, while this work addresses three specific domains (i.e. Social testing, outage modelling simulation and chat segmentation boundary classification) in computer science. We demonstrate that with the right analysis toolkit, drawing on the wealth of statistical techniques developed over the past three hundred years (i.e. Data modelling and Linear regression), converting data to information is possible.

Field defects are a useful metric to understand the quality of a given product or service. The orthodoxy states that software should be as bug-free as possible before releasing to market. While this was true of software delivered using protracted development cycles (Waterfall), this may not be true in the era of CD. Through analysis of defect data, it is essential to understand the role the customer plays in field defect discovery. Chapter 3 address this requirement.

Cloud outages are seen as the most severe type of defect, given the central point of failure. In other words, if a Cloud infrastructure is unavailable, a software vendor may lose vital revenue from downtime. Therefore it is crucial that analysis is conducted on both the time between outage events and the time to service such events. The work involving these two case studies became chapter 4.

Modelling the inter-arrival and service times of Cloud outages appeared a useful exercise in its own right. However, for small teams, being able to schedule team planning around future Cloud outages became a logical conclusion of this work. Chapter 5 looks at the modelling result of outage inter-arrival and service times and combining with a simple queue model to predict outage busy times.

From research into Cloud outage events, it became clear that DevOps teams, use real-time chat applications to discuss, diagnose and resolve outage events. We believed that the duration of these discussions would provide a useful analogue to Cloud outage service times. With this in mind Chapter 6 turned our attending to modelling real-time chat discourse conversations.

Through the modelling research conducted in chapter 6, we extended our chat discourse analysis first to investigate techniques to enhance topic modelling to improved understanding and readability of such outputs. Additionally, we wanted to close this body of work by looking at chat boundaries. Was it possible to train a computer to detect boundaries within conversations? Chapter 7 contains this work.

We acknowledge that a great number of studies and prior research has been conducted in the realm of field defect analysis, Cloud outage events, service time modelling and in the field of real-time chat discourse segmentation. However we note that a great number of these studies have been conducted as part of prior work using either a Waterfall or Agile development model. Furthermore, we note that discourse segmentation has been primarily conducted on twitter social discourse.

Recalling the research questions from section 1.3, this thesis aims to add to the existing literature as follows:

- What types of field defects do customers find once software has been released into the field as part of a rapid release delivery model? The result of this question will add to the existing published work for both Waterfall and Agile software delivery.
- How many field defects are located within the first four weeks of a release to the field as part of a rapid release delivery model? The result of this question will add to current literature in the realm of non CD/CI release models.
- What type of probability distribution can be used to model the inter-arrival time between Cloud outage events? Our result aims to add to the existing literature in the realm of non-Cloud outage events.

- What type of probability distribution can be used to model the service times of Cloud outage events? Our service time distribution result for Cloud outages can add to the wealth of work in the realm of repair times of computing systems.

- Can a simple queue model simulate Cloud outages with a degree of precision similar to observed outage data? This work will provide a useful addition to resource management modelling.

- By segmenting chat discourse using burst and reflection periods, can more words be made available to topic model software? The result of this work adds to the existing social discourse literature by focusing on real-time chat discourse rather than twitter postings.

- Can we use a supervised machine learning algorithm to detect the boundaries? The result of this final piece of work adds to the body of work in the area of topic drift identification and conversation boundaries within real-time chat discourse.

CHAPTER 3

Social Testing

Continuous delivery (CD) and delivering software for the Cloud represents a challenge for small software test teams, because of the continuous introduction of new features and feedback from customers. In this chapter, we examined two datasets in the form of two case studies. In the first, users are encouraged to report defects through social or other incentive schemes. In the second, we consider field defect detection rates in a framework where these rates are used to refocus in-house test resources. Using two enterprise datasets, we address the following questions: what types of defects can best be found in the field, allowing in-house test resources to be refocused? How soon after a system goes live are defects detected? In our first study, we show that 64.3% of the errors found by the customer are Functional defects of minor severity. In our second study, we demonstrate that once a Cloud release goes live, 31% of all Field defects are found in the first week of a release. The benefit to small test teams is two-fold: minimise the number of defects found in the field by maximising internal usage through 'Dogfood' programs, by leveraging crowdsourced test methodologies and by adopting a reward scheme to incentivise customers to find low severity field defects.

3.1 Introduction

SMEs are the backbone of the European economy, in 2015 a little under 23 million SMEs in the non-financial business sector generated 3.9 trillion Euro of value added and employed 90 million people [143]. The European customer is

maturing technologically and now demands more from their interaction with products and services. This has placed an additional challenge on the SME to provide products and services in a rapid and connected way. Nine out of ten SMEs in Europe have less than ten employees [144] in the non-financial business sector which, makes it difficult for these micro-enterprises to find the necessary additional capacity to cater for the new European customer. European SMEs are beginning to see an increase in economic growth. A 5.6% increase in SME economic activity was recorded in 2015 [143]. Additionally SME employment growth grew by 1.5% [143]. As the economic recovery continues, SMEs are looking to maximise their potential.

CD is seen as one approach that can be readily adopted by the SME to help reduce the software delivery lifecycle. CD promotes the faster delivery of software components, features and fixes [3]. With an accelerated delivery of product/service improvements, SMEs want to keep pace with large enterprise solution providers. In the race to provide solutions in a dynamic, agile way, large enterprises have the resources to exploit CD. These same enterprises can also leverage fully mature software test teams to ensure a succession of stable releases for the consumer and reduce the risk of subsequent brand damage due to releasing a poor quality product or service. The adoption of CD is non-trivial. Recent work has been conducted to outline the key challenges faced by software companies. These include the development of features rather than components and the development of an automated test harness to support testing [145]. An SME cannot compete at this level.

Cloud computing is seen as a way for SMEs to compete with larger companies, regarding the rapid delivery of software and services. This is due in part to the 'always on' nature of Cloud computing. However, both micro-teams and SMEs face continuing challenges when adopting both Cloud computing and CD as their hosting and software delivery mechanisms respectively. A recent study has outlined the issues facing SMEs in adopting industry standards concerning software development and delivery [146]. Therefore, in this chapter, we analyse the factors that may impede the rapid delivery of high-quality software for teams with low levels of resourcing.

At regular intervals, throughout the past twenty years, software quality commentators have discussed the need for companies to extensively use the soft-

ware they develop before releasing to the customer. This practice is known as "Eating your own dogfood" [147]. A recent article in Forbes by the CEO of Lua (a startup company) discussed the need for companies to continuously use their software before release [148]. A leaked memo from the CEO of Yahoo lamented the fact that only 25% of employees were willing to use Yahoo mail as their corporate e-mail reader [149].

In this chapter, a framework is proposed that both micro teams and SMEs can use to deliver high-quality software to the Cloud while utilising their limited resource cohort. The core idea of this framework is two-fold: (1) for the in-house test teams to focus on high-value test areas, while incentivising the customer to find low impact field defects and (2) for software test teams to amalgamate based on the types of defects found coupled with regular internal use of their own software and additional crowdsourced test methods. For micro teams with a limited pool of test resources, leveraging a crowdsourced team of tests resources could aid defect detection.

This chapter contains details of two case studies that were conducted on a large enterprise dataset of both in-house and field defects. Through this study of customer defect data, we show what types defects the customer is useful at finding and is there an overlap between the types of defects that in-house test teams discover.

By leveraging the customer's skill at finding certain categories of defects, we suggest incentivising the customer to find a particular class of defect. This may aid the SME to deliver higher quality software by diverting in-house test resources to high-value areas such as performance and systems testing. Additionally using the results of this study for our framework, a crowd-sourced dogfood program and an in-house test team alignment can be used to reduce the number of defects found in the field.

3.2 Case study 1 - Social testing

Defect studies have been shown to provide an effective way to highlight customer usage patterns of software. Defect studies can also aid businesses to align their test coverage more towards customer based use cases.

The research presented in this case study examines 1394 field defects from a

large enterprise, Cloud-based system. The data was collected over a 12-month period (Jan 2015 - Dec 2015) and is comprised of four main components: E-mail, Collaboration, Social and Business Support System (BSS). The systems have been deployed within three data centres and are used by customers globally. The software is developed in Java and runs on Linux. Product development follows a CD model whereby small amounts of functionality are released to the public on a monthly basis. For each defect, we have access to the full defect report, but we particularly focus on the defect impact, defect component, data centre location and defect type.

This case study aims to answer a number of questions. First, How do field defects impact the customer's overall user experience? Second, what components are likely to yield field defects? Third, what data centres are likely to yield more field defects? Finally what types of defects do customers typically find?

To answer these four questions, this study is broken down into the following attributes: defect impact, defect component, data centre location and defect type.

3.2.1 Defect Impact

A loss of functionality at either a system or client level is categorised as critical, major or minor. The following defect severity guidelines provided are defined by the enterprise that created the data set. A critical defect can be defined as a defect where there is a loss of core functionality from either a server-side component or from a client-side perspective. A major defect can be defined as a defect where there is some loss of functionality, but the loss is not system-wide nor does the loss affect all end users. A minor severity defect can be defined as a defect with no loss of data, but some form of unexpected behaviour has occurred. Other ways in which the impact of the defect can be expressed is by the number of customers, who experience the same type of problem. Finally, it should be noted that defects of a similar type can vary in impact depending on whether they were raised as an in-house or field defect.

3.2.2 Defect component

Understanding the location of field defects at a component level gives an awareness of how customers use the product and more importantly what types of

defects they are useful at finding. For example, in-house test teams may design a set of tests, which will find a certain class of defect. Field defects can provide test teams with insight as to potential gaps in their coverage. Depending on the nature of these test gaps and the size of the test organisation, they may be difficult to close. For this study, we categorised software components as follows: e-mail, collaboration, social and BSS.

3.2.3 Data centre location

Understanding the location of field defects at a data centre level can highlight whether a specific data centre or high usage is a factor in the number of field defects raised. There are three data centres in our dataset: data centre A (High usage), data centre B (Low usage) and data centre C (Medium usage).

3.2.4 Defect type

We consider three defect types: Functional, Performance and System. The following defect type guidelines provided are defined by the enterprise that created the data set.

A functional issue may relate to the behaviour observed directly by the customer, for example, a component feature when used may either fail nor work entirely as expected.

Performance defects fall into two main categories, client-side and server-side. For client-side issues, an end user may experience an unresponsive or slow UI. Additionally, a server-side performance defect may be related to a sudden burst of user activity, which has an undesirable performance impact for the entire system.

System defects relate to a class of problem where either an end-to-end system workflow has failed. Or by having multiple concurrent users using the system at a given point in time has caused a feature or process to fail.

3.2.5 Limitations of dataset

The dataset has a number of practical limitations, which are now discussed. Defect severity can vary depending on the support engineer filing the bug report or the customer logging the field defect. This subjectivity can lead to a different severity rating being assigned to the same type of defect.

Figure 3.1: % Field defects by severity

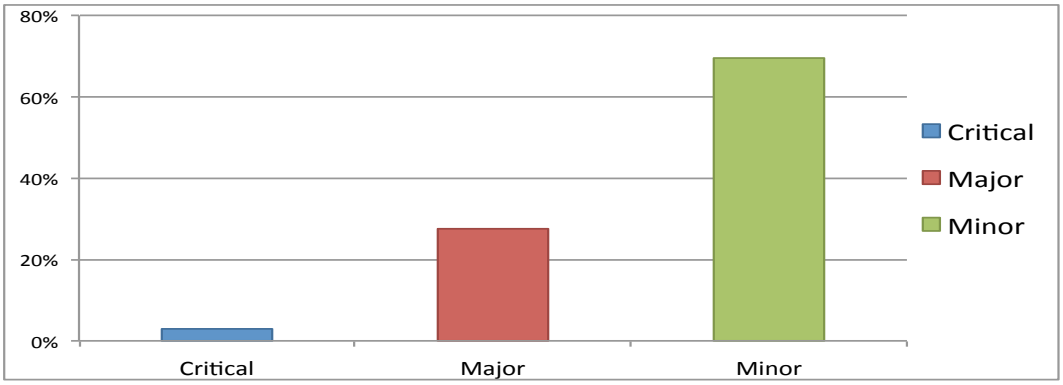


Table 3.1: % Field defects by severity

Severity	Critical	Major	Minor
% of Total	2.9%	27.5%	69.5%

While the field defect tracking application has a granular system to aid the classification by functional location, there are challenges in locating the parent area of a defect particularly when the defect displays errors in multiple sub-systems. The severity and the functional location of each defect was reviewed for this case study. The goal was to ensure that each defect regarding severity and categorisation remained constant.

The defects that form part of this study are from a large enterprise Cloud system. The defects apply to the domain of e-mail, collaboration, social and BSS. Our findings may be less applicable to other domains.

3.2.6 Results - defect impact

Figure 3.1 shows the percentage of the total defects broken down by severity. Minor defects are the most common with critical defects being the least common.

Field defects were classified by impact, which is shown graphically in Figure 3.1 and textually in Table 3.1. These show the percentage of all defects of each severity type. The majority of defects found by the customer had a minor impact on their user experience (approximately 70%), while approximately

Figure 3.2: % Field defects by component and severity

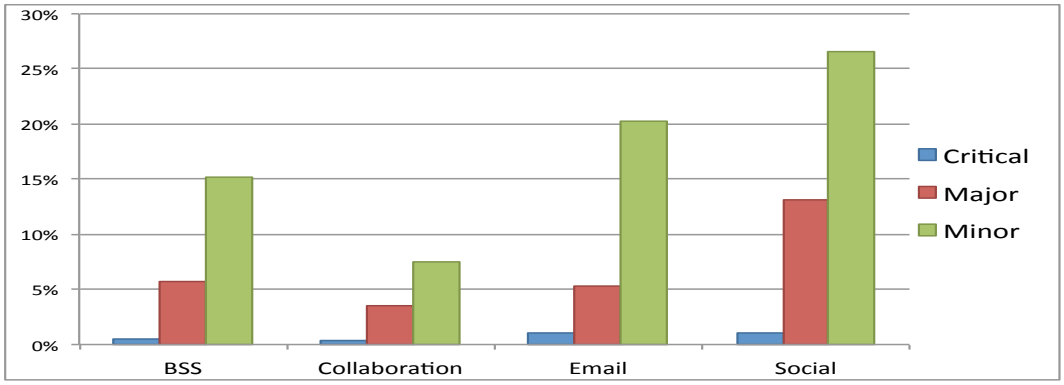


Table 3.2: % Field defects by component and severity

Severity	Critical	Major	Minor	Total
BSS	0.5%	5.7%	15.2%	21.4%
Collaboration	0.4%	3.5%	7.5%	11.4%
E-mail	1.1%	5.2%	20.2%	26.5%
Social	1.0%	13.1%	26.5%	40.6%

28% of users experienced a major severity defect and the remaining defects (just under 3%) were of critical severity.

3.2.7 Results - defect component

Figure 3.2 shows the percentage of the total defects broken down by component and their severity. In each component, minor defects are the most common with critical defects being the least common.

Table 3.2 shows the percentage of all field defects broken down by component and severity. The Social application contained the most defects (41%), E-mail (27%) and BSS (21%) had a broadly similar level of defects, and while the collaboration application had the least percentage number of defects found with 11%.

3.2.8 Results - data centre location

Figure 3.3 shows the percentage of the total defects broken down by data centre and severity. In each data centre, minor defects are the most common

Figure 3.3: % Field defects by data centre and severity

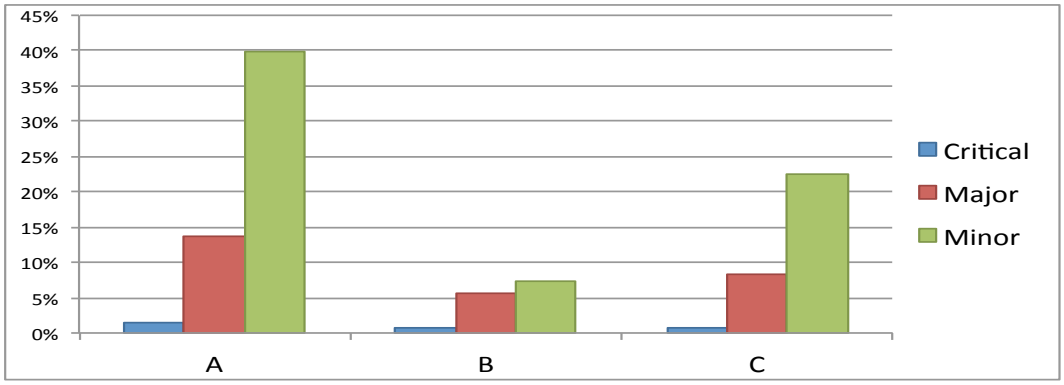


Table 3.3: % Field defects by data centre and severity

Data Centre	Critical	Major	Minor	Total
A	1.6%	13.6%	39.8%	55.0%
B	0.7%	5.6%	7.3%	13.6%
C	0.6%	8.3%	22.4%	31.3%

with critical defects being the least common. Previously it was noted that both centres A and C are high and medium usage, while data centre B is low usage. Given the level of field defects found in each data centre, this supports the intuition that higher usage leads to a greater number of defects.

Table 3.3 breaks down the Field defects by data centre and by severity. 55% of all field defects found were in data centre A. Data centre C recorded 31% of defects, while data centre B recorded only 14% of defects.

3.2.9 Results - defect type

Figure 3.4 shows the percentage of the total defects broken down by testing type and severity. It was expected that minor severity would feature significantly, it's interesting to observe that the majority of functional defects are minor. It was observed that for defects classified as System that the number of major and minor defects are practically the same. Finally, it was noted that the customer found few Performance defects. However, of the defects that were found, slightly more were major than minor severity.

Figure 3.4: % Field defects by test type and severity

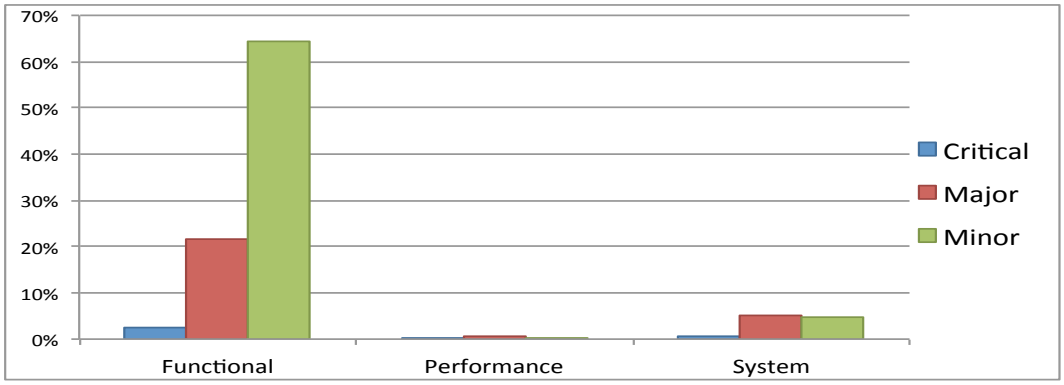


Table 3.4: % Field defects by defect type and severity

Field Defect Type	Critical	Major	Minor	Total
Functional	2.4%	21.8%	64.3%	88.5%
Performance	0.1%	0.7%	0.4%	1.2%
System	0.5%	5.0%	4.7%	10.3%

Table 3.4 shows the percentage of field defects found according to their type. 89% of all customer issues reported were functional, a further 10% of customer defects were related to end to end system issues, while only 1% of defects found were related to the performance of either client interface or underlying server system.

3.2.10 Discussion - defect impact

To answer the question *how do field defects impact the customer's overall user experience?*, Figure 3.1 and Table 3.1, show that the customer finds more minor defects than any other type, almost 70%. This means that the customer will come across some unexpected behaviour which does not result in data loss during their day to day product usage. Given that minor severity defects are found most often, the logical conclusion is that these types of defects are found by the customer exercising the most common component use cases. With the level of major defects found being 28%, this also shows that these defects were found as part of a typical customers day to day usage, albeit to a lesser degree than minor defects. Interestingly, as part of the customer's typical

use case, they did encounter some form of data loss or non-trivial unexpected behaviour. With approximately 3% of field defects being critical, this suggests that the likelihood of a customer experiencing a total component or some system-wide failure is a rare event. That said customers, either by themselves or in conjunction with other users, were able to bring about behaviour, which impacted the wider system.

Given the nature of CD, it is important to note: new code and features are released frequently. Therefore, new field defects are raised on a continuous basis once new features are delivered. If a bug bounty program were introduced to incentivise field defect discovery, it would be interesting to determine the increase in the velocity of field defects from the introduction of such a scheme. Typically bug bounties have been the preserve of security defect discovery. By rolling out such a scheme for field defect discovery in general, it would be valuable to both the software developer, to focus on testing code paths which lead to higher severity issues, while incentivising the customer to uncover lower priority defects.

3.2.11 Discussion - defect component

Examining defect component indicates where customers are likely to find defects within each component.

Figure 3.2 and Table 3.2 highlight that, at a component level approximately 41% of the field defects found were in the social component. A further 27% found in the e-mail component, with 21% in the BSS component with a final 11% found in the collaboration component. While defect yields may not map directly to application usage (unfortunately, no application usage metrics were available), conditional probabilities were calculated to determine the likelihood of certain combinations of defect attributes being found. With minor field defects being raised most often, conditional probabilities for each component at minor impact level were calculated. It was noted that P(minor|e-mail) had the highest probability with 0.762, P(minor|BSS) with 0.709, P(minor|collab) with 0.660 and P(minor|social) with 0.654. This tells us that the customer is more likely to find a minor impact field defect in the e-mail component.

This may seem counter-intuitive given that 41% of field defects were found in social. We concluded that given the lower level of major impact defects

found in e-mail increases the likelihood of a minor impact defect being found. The logical conclusion is that the more a component is used, the more defects are likely to be found by end users. We suggest that for popular components (in our study e-mail & social) that have continuous feature releases, by incentivising the discovery of lower impact defects by the customer, can help in-house testing refocus their efforts on the major severity testing across all test disciplines in their key feature/components areas.

3.2.12 Discussion - data centre location

Figure 3.3 and Table 3.3 give an insight into field defect breakdown by data centre. As mentioned previously, it is known that the level of usage varies from data centre to data centre, interestingly the customers of data centre A (High Usage) reported the highest number of field defects with 55% while data centre C (Medium Usage) and data centre B (Low Usage) had 31% and 14% defects raised respectively. There may be some form of correlation between concurrent user population and field defects raised.

Checking conditional probabilities for each data centre for both minor and major defects, as follows P(Minor|DC-A) and P(Major|DC-A) gives 0.724 and 0.248 respectively. These conditionals state that the customer is more likely to find a minor defect within data centre A. For data centre B the following conditionals were calculated; P(Minor|DC-B) and P(Major|DC-B), which gives 0.537 and 0.411 respectively. These conditionals tell a similar story to that of data centre A, that the customer is more likely to find a minor defect than a major one. It is conjectured that the customer use case on data centre B is different to that of the other two data centres. Further analysis should be employed by the in-house test teams, to ensure their test scripts cover the main customer use case in data centre B which generates major impact field defects. Finally checking P(Minor|DC-C), P(Major|DC-C) gives 0.714 and 0.265 respectively. These conditional probabilities are very similar to those of data centre A. Customers are almost three times as likely to encounter a minor impact defect on data centre C than that of a major impact field defect.

Overall we found that for high and medium usage data centres the likelihood of finding minor field defects was almost three times that of finding a major impact defect. For the low usage data centre, the probability of finding a major impact field defect was broadly similar to that of finding a minor impact field

defect. Finally, the customer was less effective at finding high severity defects irrespective of the data centre used.

In the context of CD, the same code is released to each data centre; customers are more likely to be impacted differently depending on data centre. Knowing the underlying customer data centre use case is key. With knowledge of both data centre usage and field defect data, incentivisation schemes can be tailor-made according to each data centre. One suggestion would be a bounty to target minor field defects on high usage data centres, while refocusing in-house resources to find more major impact defects before release.

3.2.13 Discussion - defect type

Finally, to understand which type of field defect it is that the customer typically finds, defect type was examined. This metric may be one of the most important regarding this study. It helps underscore which class of defect the customer is proficient at finding.

Figure 3.4 and Table 3.4 indicate that functional defects are the most commonly uncovered by the customer with 89% of all defects found being functional. Also of significance is the severity of these defects with 64% being minor severity. Functional defects typically present themselves in the form of user experience behaviour errors where the end user attempted an operation and the behaviour encountered was unexpected. It is also important to note that 10% of all issues were system errors, typically these manifest themselves as unexpected behaviour during active concurrent usage. One can infer that system errors are less common than functional ones. It may also be the case that system errors do not readily manifest themselves to the end user in the same way as functional defects.

Performance defects ranked the lowest in overall defects found with only 1% of all problems being attributed to performance defects. This data suggests that either the performance of each component was adequately tested before release or that performance defects may be harder for the end user to measure and quantify once in the field.

From a customer's perspective, they are more likely to find functional defects as these issues are found within the UI. However, the customer finds a greater proportion of minor severity functional defects. Additionally, the customer

was less effective at finding critical severity Performance and System field defects. From a CD/CI perspective, a balance needs to be struck concerning the features being released and their likely defect type yield. For backend server features in-house test teams can focus almost exclusively on Systems and Performance testing. For features with rich functionality, in-house test teams can focus on use cases, which are likely to yield critical and major defects with some additional minor impact areas.

Regarding bounties, for releases with high functional content, software developers could award triple/double and single prizes for critical, major and minor impact field defects respectively with the knowledge that adequate testing was conducted in-house for critical and major use cases. Similarly, for releases with high System and Performance and low functional features, proportional bounties may be awarded.

3.3 Case study 2 - Eat your own dogfood

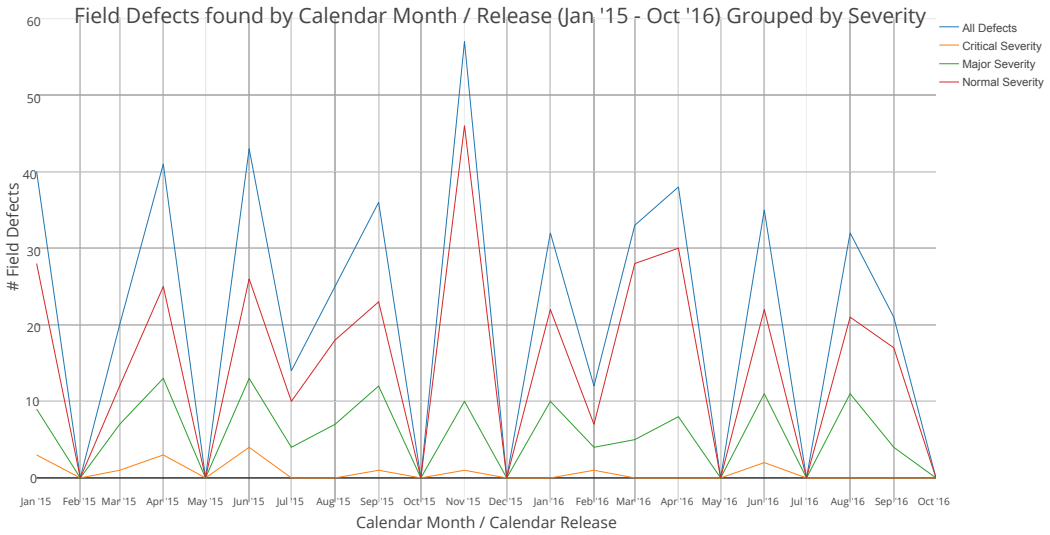
Field defect studies have been shown to provide a useful way to infer gaps within in-house testing [150][151]. Such studies, in conjunction with our framework, can be adopted by micro teams and SMEs to determine common failure types.

The research presented in this second case study examines 2008 defects (both field and in-house) from a large Cloud-based real-time collaboration system. The data was collected over a 22-month period (Jan '15 – Oct '16) and is comprised of three main components: Instant Messaging, Web Conferencing and an interactive audio and video component.

Figure 3.5 shows both the total number of field defects found and the total critical, major and normal defects found post release of each of the 16 releases over a 22-month period (Jan '15 – Oct '16). We note that there were no releases during the following months: (February, May, October December (2015) and May, July and October (2016)).

There are four in-house test teams whose function are to find software defects in the categories Function test, Performance test, Security and System test. Defects are also found by Development, DevOps, Support, Accessibility and well as other general users. Field defects are found by either the customer

Figure 3.5: Field Defects found by Calendar Month / Release (Jan '15 - Oct '16) Grouped by Severity



or any of the in-house teams just mentioned. Defects are of the following four types: Functional, Performance, Security and System. Defect severity is categorised as either: Critical, Major or Minor. The systems have been deployed within three data centres and are used by customers globally. The software is developed in Java and runs on Linux. Each release takes place on a Saturday.

Product development follows a CD model whereby small amounts of functionality are typically released to the public approximately every four to six weeks. For each defect, we have access to the full report, but we particularly focus on the defect severity, defect type and found by whom.

This study aims to answer the following questions. First, what group is most likely to find either an in-house or field defect based on defect severity and test type? Second, what is the field defect discovery rate during the first fourteen days of a release? To answer these questions, our study is divided into the following two subsections: defect discovery probability by team and field defect discovery rates.

3.3.1 Defect discovery probability by team

A question to software companies is, *what types of defect are found both internally and externally*? Similarly, *what is the most common severity types found in-house and in the field*? By analysing the types of defects found both internally and externally, a map can be built to determine which teams are most effective at finding a particular class of defect. Likewise for defects found in the field a similar map can be drawn to determine what types of defects a customer is good at finding. Given the limited resources of both micro teams and SMEs, an intersection of both maps could be used by internal test organisations to a) realign their test organisation to discover more defects and b) pivot internal test practices to more customer based usage patterns.

3.3.2 14/28 days later – defect discovery rates

Studying when defects are found in the field gives us knowledge of how reliable software is. In the field of system reliability, there is the idea that failures (defects) may follow a specific pattern which can be represented in the form of a 'bathtub curve' [152] [153]. The patterns can be summarised as follows: 1) Decreasing failure rate (early failures), 2) Constant failure rate (random failures) and 3) Increasing failure rate (wear-out failures). Of interest is to understand if field defect failures found within the first twenty-eight days conform to these characteristics. Of additional interest is to understand what types of defect (and their associated severity) occur within the first two weeks of a release.

3.3.3 Limitations of dataset

The dataset has a number of practical limitations, which are now discussed. While the defect tracking system allows for a granular categorisation system, whereby field defects can be mapped to a specific release. There were a number of field defects that were mapped to an incorrect release. The authors used the defect creation date to determine which field defects belonged to which release.

The defect reports that form part of this study are from an enterprise Cloud system. As a result, the analysis may not be relevant outside of these fields.

Figure 3.6: In-house defect severity detection probability By Team

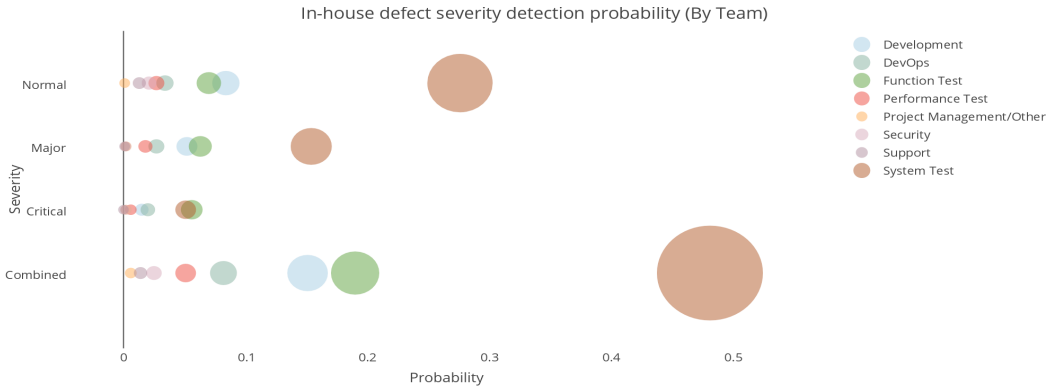
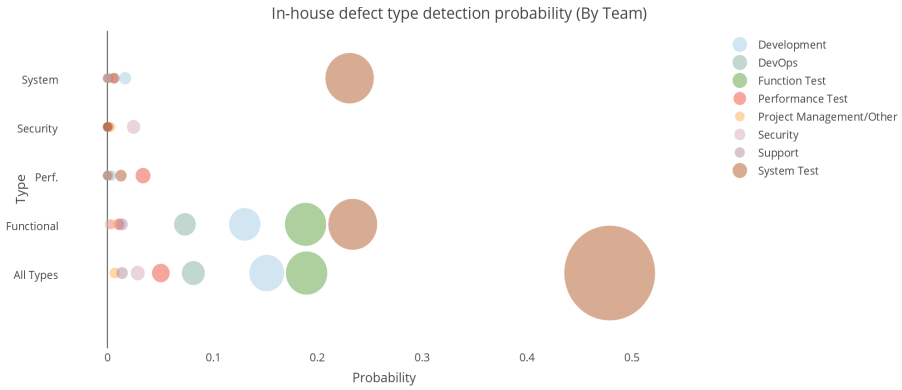


Figure 3.7: In-house defect type detection probability (By Team)



3.3.4 Results - defect discovery probability (By Team)

Figure 3.6 shows a bubble plot of in-house defect detection probability grouped by team type and defect severity. The size of the bubbles is scaled relative to the probability. The greater the probability, the larger the bubble diameter. We also show a probability of combined defect severity. The System test team have the highest probability of finding a normal or major defect. System test is also most likely to find a defect of any severity. Function test is most likely to find critical defects.

Figure 3.7 shows a bubble plot of in-house defect detection probability grouped by team type and defect type. We also show the probability of all defect types. The System test team have the highest probability of finding a System, Functional and combined defect type. The Performance team are most likely to find a performance defect. Likewise, the Security team are most adept at find-

Figure 3.8: Field defect type detection probability (By Team)

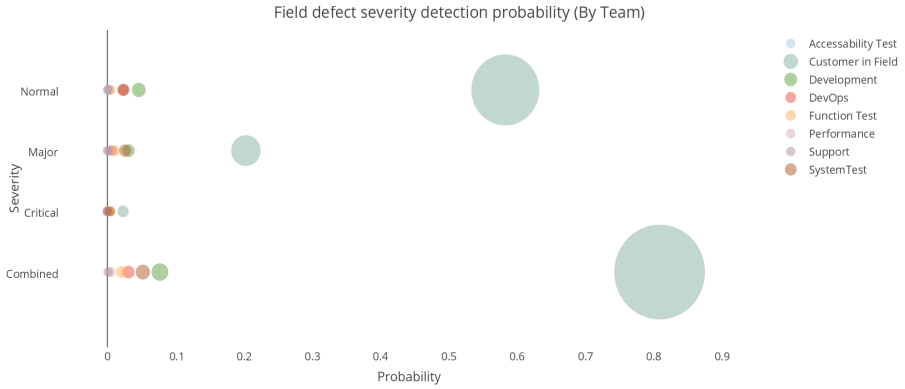
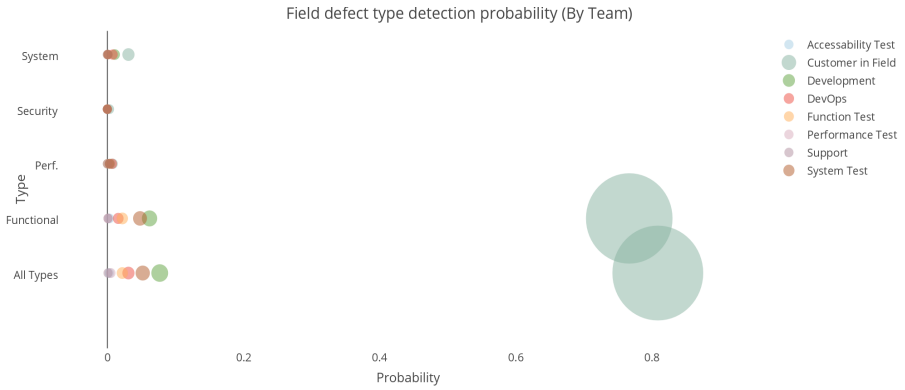


Figure 3.9: Field defect type detection probability (By Team)



ing security defects. As hoped, individual teams have the highest probability of finding team centric defects, with the exception of the Functional team.

Figure 3.8 shows a bubble plot of field defect detection probability grouped by team type and defect severity. We also show a probability of combined defect severity. The customer is most likely to find a defect of any given severity in the field.

Figure 3.9 shows a bubble plot of field defect detection probability grouped by team type and defect type. We also show a probability of all defect types. The Customer is the most likely group to find a defect of any given severity in the field.

Figure 3.10: % aggregate field defect detection rate (First 14 days of a release)

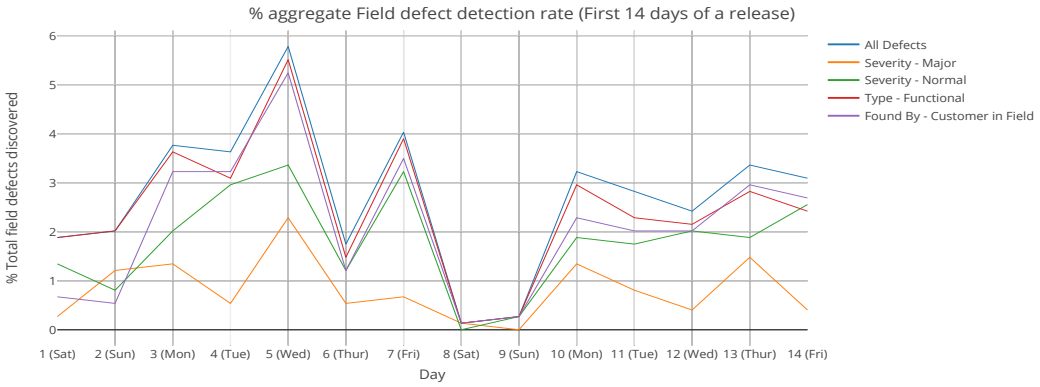
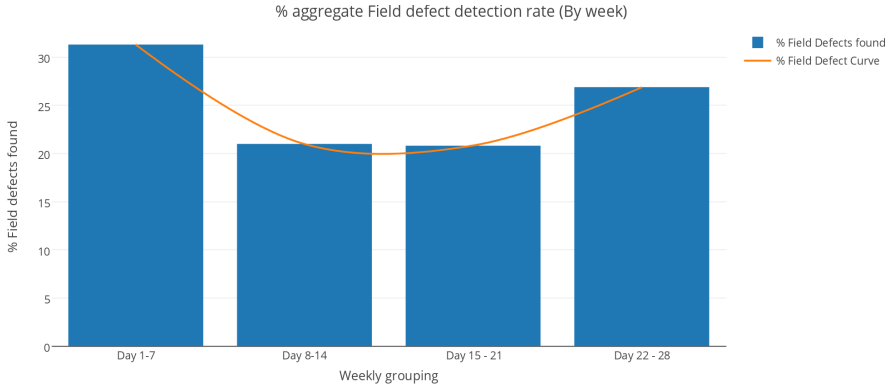


Figure 3.11: % aggregate field defect detection rate (By week)



3.3.5 Results - 14/28 days later – defect discovery rates

Figure 3.10 shows a line plot of percentage field defects raised during the first fourteen days of a release. The percentage values are an aggregate over the entire fifteen releases studied. Additionally, the percentages are calculated as follows: an aggregate daily rate divided by the total number of field defects found. Days five and seven saw the highest aggregate percentage of field defects raised, while days eight and nine saw the lowest aggregate percentage field defects detected. We note that days eight and nine are weekend days (Saturday and Sunday).

Figure 3.11 shows a bar plot with a fitted curve of percentage field defects raised during the first four weeks of a release. The percentage values are an

aggregate over the entire fifteen releases studied divided by the total number of field defects found in the first twenty-eight days of a release. Week one saw the highest number of field defects raised with just over 31%, while weeks two and three had near identical rates with approximately 21%, while week four had an increased rate (over weeks two and three) of almost 27%.

3.3.6 Discussion - Defect discovery probability by team

As mentioned in our case study introduction, we wanted to understand, for in-house defects, how defect detection was distributed across each internal test team by defect severity and type. Interestingly, for defect type, Figure 3.7 illustrates that the System test team were most likely to find System and Functional defects (both 0.23), while the Performance and Security teams found the most performance and security-related defects. Overall when all defects types are reduced to a single category the System test team are most likely to find a defect irrespective of type (0.48).

Figure 3.8 shows that the customer is most likely to find a defect in the field, irrespective of severity (0.81). It's worth noting that for normal severity defects the customer has a probability of 0.58 while the next highest is the development team with a probability of 0.05.

Figure 3.9 highlights that the customer is highly skilled in detecting functional defects (0.77) and has a low probability of finding other types of defects: Performance (0.01), Security (0) and System (0.03). For non-functional defects, internal consumers (i.e. test teams) do find "field defects" as part of their daily usage of the software. Development (0.01 System defects) and DevOps (0.01 Performance and System defect types).

A number of interesting points are raised by the analysis of both sets of defect data. Firstly that the *System test team have the highest probability of finding a defect in-house when severity or type are reduced to a single category.* What is surprising though is that while the *Performance, Security and System teams are most adept at finding homogeneous defect types*, the *Function test team is less likely to find a Functional defect (0.19) compared to System Test.* It is worth noting that the Development team are also quite skilled at flushing out functional defects too (0.13). Second that the Customer is more likely to find

a field defect than internal users "Dogfooding" their own software. That said, the customer appears to find a very specific type of defect, a normal severity functional defect.

Given the overlap, concerning detection of Functional defects, there is an argument to merge both Function and System test teams into a single group. Based on the dataset we know that the System test team are skilled at finding functional defects, by merging teams there would be the added benefit of upskilling the functional team members to test and detect System defects. Furthermore, it makes sense to have a separate team to test both the Performance and Security areas of the product because this testing requires specialist knowledge which enables these test teams to focus on their core areas of expertise.

With the customer being so adept at flushing out normal severity functional defects, focus instead should be placed on determining what test paths should be included in future test cases to capture these classes of functional defects as part of an internal testing. However, the customer should be incentivised in some manner for the number of lower severity defects they find. We discuss this subject matter in more detail in prior work [154] and in our literature review chapter 2.

Finally, irrespective of the dataset and the results tied directly to it, we suggest the following outcomes for our framework: a) test teams should be aligned based on the types of defects they find, b) in-house testing should prioritise test paths to areas of the product, which are both critical path and where the customer is least likely to find a defect and c) schemes to incentivise the customer to find low severity defects could be introduced.

3.3.7 Discussion - 14/28 days later – defect discovery rates

Examining the field defect discovery rate during the first twenty-eight days of release helps us to understand what types of defects a customer is likely to find.

Figure 3.10 shows the percentage of defects found the first fourteen days of a release. Looking at the 'All Defects' line initially we can see that field defect detection peaks at day five. Approximately 6% of all field defects (raised in

the first twenty-eight days of a release) were found on the fifth day of a release. The second highest daily rate appears on day seven with 4%. Also of note is that, on days eight and nine, we observed the least number of field defects raised (0.13% & 0.27% respectively). This may be attributed to both of these days fall on a weekend. That said the first two days of a release also fall on a weekend and see a higher rate of field defects raised (Approximately 2% on both days). Finally, as part of the second working week of a release, the rate rises to approximately 3% and remains steady at this rate for the remainder of the second week.

Another point of interest is evident in Figure 3.10. If we consider Normal & Major severity, Functional type and Customer found field defects we can see how close these lines mirror the overall rate line. This confirms our intuition that these categories of field defect are highly correlated as part of the overall data set. Our reason for this belief is that these four categories of defect contribute a significant proportion to the overall dataset. No formal regression analysis has been conducted to confirm our intuition.

Figure 3.11 presents a bar plot with fitted spline curve. This plot illustrates the percentage of field defects raised weekly during the first twenty-eight days of a release. We made the choice to aggregate the field defects by whole week. This avoids having to adjust for weekday vs weekend. We can see that the highest percentage of field defects detected within the first month of a release are found in the first week of a release. 31% of all field defects found in the first month of a release are found during the first seven days. Also of note is the rate drop during weeks two and three of release, approximately 21% for both weeks. Finally, it is worth noting that the rate increases to approximately 27% during week four.

Looking through the lens of analysis from both a fortnightly and monthly perspective, we can see that highest percentage of field defects are found within the first seven days of a release (specifically day five). There may be a host of reasons why the customer finds so many defects in such early stage of a release: Poor customer use case profiling, lack of test automation, lack of test resources. Irrespective of the root causes it is worth noting that the majority of field defects the customer uncovers are low severity and functional. Therefore a targeted set of crowdsourced tests would be useful in flushing our additional

defects before public release.

We previously mentioned the idea of software reliability and how the 'bathtub curve' is used (at a high level) to illustrate the three stages of system reliability. We observed in Figure 3.11 the weekly percentage rate of field defect discovery. We noted the high initial rate in week one, the drop in weeks two and three and finally the increased rate in week four. This behaviour can be mapped directly to the early, random and wear-out (i.e. failure that occur due to sustained usage) failures, which characterise reliability. Targeted survival analysis of field defects found in the first month of a release is required to determine whether a field defect does indeed contain early, random and wear-out attributes. We temper this finding with the observation that there is a baseline failure rate of at least 20% on any given week. Our datasets also provide evidence supporting a bathtub-type curve for software releases in a Cloud environment.

Generally speaking, companies want to see a return on investment in the shortest time possible. We can see from our analysis that 52% of the total field defects found post-release, were located in the first two weeks of a release going live. We suggest that for companies to maximise their defect detection within the shortest time possible, a two week crowdsourced–dogfood test program is appropriate. Additionally, from a scheduling perspective, a two-week window may align well with an internal development sprint cycle.

By leveraging the power of both the internal workforce and a crowdsourced test cohort for a fixed duration, companies can uncover many 'field' defects before general release. Certainly, this crowdsourced–dogfood program will uncover many defects with early and random characteristics. Based on the analysis of in-house test data, these types of defect are difficult to uncover as part of internal testing. For defects with wear-out attributes by adopting a framework of survival analysis, teams can determine if there are specific components which wear out more quickly than others. With this knowledge development, teams can adopt remediation plans to ensure their components are more robust to wear-our failures.

Software companies [155] use early adopter programmes to solicit feedback on either new products and features. Participants are generally from a computing background. However, this is not a hard requirement for all schemes.

Generally speaking, a crowdsourced programme will be useful in the domain of consumer-based popular software (e.g. social media applications), given the pool of potential participants. We suggest that a crowdsourced programme for a niche piece of software may be more difficult to staff and implement given the reduced pool of participants compared to popular software types.

Putting the findings drawn directly from our dataset to one side, we can add the following proposals to our framework: a) a period of crowdsourced-dogfood testing should be conducted for a two week period before release, b) the crowdsourced team should contain individuals from a variety of backgrounds.

3.4 Conclusion

Previous studies have shown that crowdsourced methods can be an efficient way to find bugs that in-house teams and automation find difficult to detect (see section 2.4.1). Additionally, bug bounty rewards can provide an incentive to end users to improve software quality once in the field (see section 2.4.2). Given that bug bounty programmes are useful in the security domain [156], consideration should be given to using in non security domains. Furthermore, that analysis of field defects is a valuable exercise (see section 2.4.4).

The purpose of the presented case studies was two-fold. First to examine the role of the customer in the generation of field defects and secondly to inspect the types of errors found by in-house teams and the customer in the field.

In both case studies, we found that the customer found many minor-severity functional defects. In our second case study, field defect detection occurs most often in the first seven days of a release and that over a monthly period field defect detection rates mirror those seen in other fields of reliability engineering.

The findings of these studies (see section 2.4.1) support previous work mainly in the gaps between in-house software testing and general customer usage. This work provides an additional study of software developed using a CD model. Adoption of CD means continuous feature releases and continuous defects, however, feature releases may be delivered in such a way to ensure that there is not a significant burden on in-house test teams.

In future, SMEs and micro teams could adopt a reward bounty scheme to customers who find low severity defects in the field. Likewise, we suggest a

co-ordinated system of crowdsourced dogfood testing could be implemented before release; the benefit would be to detect additional defects prior to release. A two-week window for crowdsourced testing may provide a balance between defect detections and improved software quality before shipping.

In our next chapter, we extend the idea of Field defect detection and focus on Cloud outage events. We look at how these events are typically triggered by regular behaviour. Additionally we model the inter-arrival time between outages and the service time to resolve such issues.

CHAPTER 4

Outage Modelling

Hosting software applications in a Cloud-based infrastructure represents challenges for micro teams and SMEs, due to the variety of ways in which production outages can occur. We consider both the inter-arrival and service (repair) times for outage events in a framework where these downtimes are used to re-focus DevOps resources. Using an enterprise dataset, we address the question of how inter-arrival and service times of outage events are distributed and what relationship the service times have with different types of failures that can occur in a Cloud data centre. If a company performed analyses such as these on their own data, it could aid understanding of their outage inter-arrival and service time distributions.

4.1 Introduction

For micro teams and SMEs, the adoption of Cloud technology is no easy task. Due to resource constraints and multiple failure patterns, small teams face challenges in providing a reliable and stable service platform for their customer's needs [32].

One way to provide services with an elevated market reach is through a Software as a Service (SaaS) model. This Cloud-based approach is seen as a shift away from highly complex bespoke solutions, to more focused and cost-effective solution [17]. As customers demand highly effective services to solve their business problems, a Cloud platform can help keep pace with these needs.

A single delivery platform is used to host multiple software solutions and services.

However, SMEs face a number of key challenges when embracing a Cloud service model, especially in the area of reliability and maintainability. Recent work has highlighted a number of challenges, which include: outage frequency and duration. Almost all SMEs (93%) employ less than ten people [157]; therefore in this chapter, we analyse the factors that may impede reliability especially for businesses with low levels of resources.

In this chapter, we describe a framework that the SME can use to best manage their limited pool of resources. The core idea of this framework is for Cloud operations teams to focus on areas with long outage service times (typically areas with high manual processes) to reduce the overall outage time. This chapter contains two studies of software outage data from a large enterprise dataset. Through the study of outage event data, we show a) how to first model the inter-arrival time of Cloud outage events, b) by modelling the service times of outages which types of outage events take the longest to resolve.

For enterprises with a more extensive set of resources and a requirement to deploy software to multiple data centres, we also consider why having standardised homogeneous data centres are crucial to reducing outage service times, and how application types play a role in the duration of outage remediation.

For businesses who provide their Cloud platform to allow companies to host services or solutions, this is known as Platform as a Service (PaaS). These providers allow for multi-tenancy. It is proposed that high-level outage data could be shared between organisations to triangulate cross-application outage events.

4.2 Case study 3 - Outage inter-arrival time modelling

The study presented in this case study examines 246 Cloud outage events from a large enterprise system. The data was collected over a 12-month period (Jan 2015 – Dec 2015) and is comprised of four main components: e-mail, collaboration, social and Business Support System (BSS). Additionally, the class of outage have been grouped into the following main categories: Configuration/-

Manual Process, Contention/Concurrency, Disaster Recovery, Network and Hardware/Other. The systems have been deployed within three data centres and are used by customers globally. The software is developed in Java and runs on Linux.

Figure 4.1: All Outages Raised Per Month During 2015 (Including Release Windows)

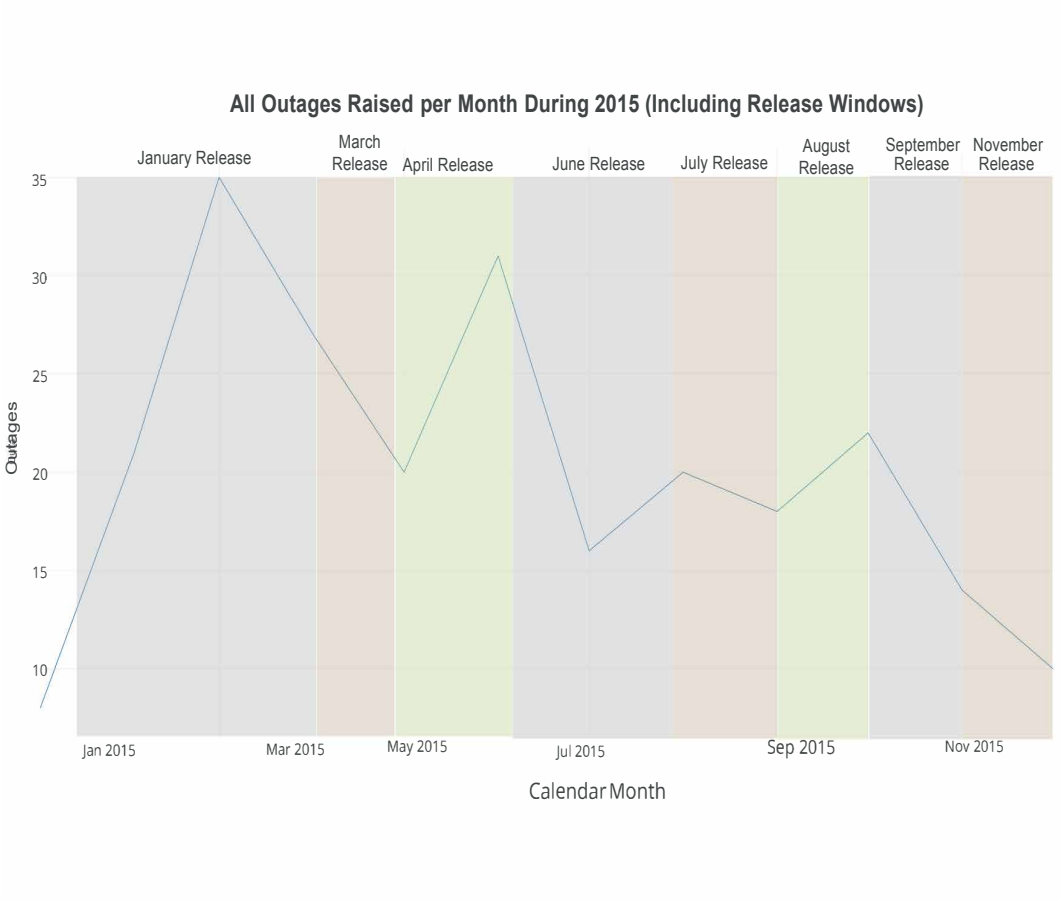
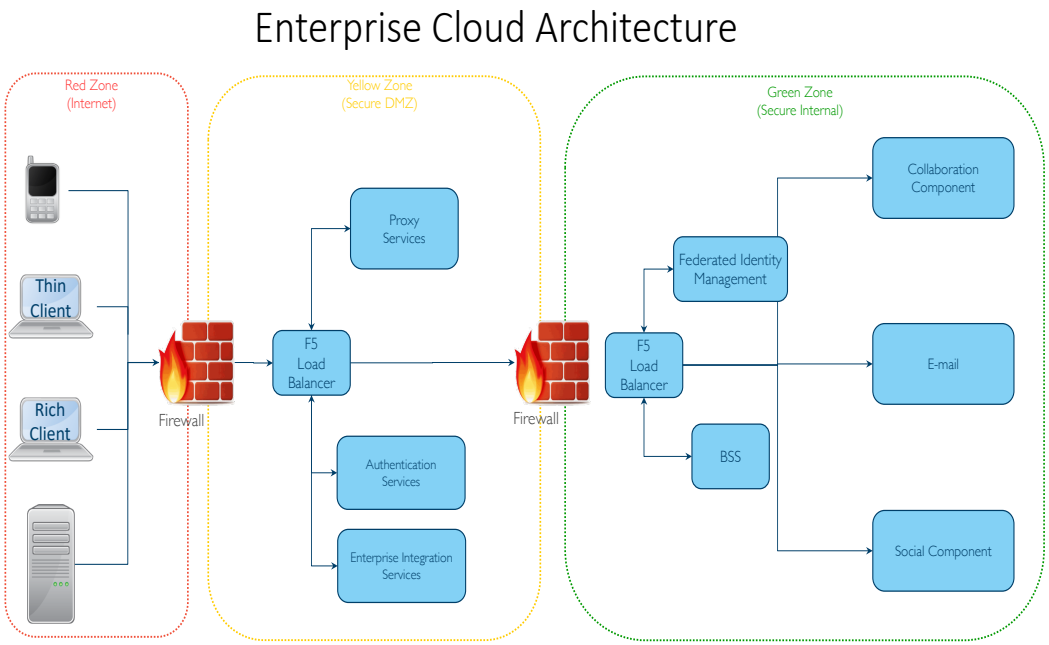


Figure 4.1 shows a time series graph of the Cloud outages raised during 2015. The graph also shows (shaded) the Cloud release windows during 2015. We note that there were eight major releases during 2015 (January, March, April, June, July, August, September and November).

Figure 4.2 shows at a high level the architecture of the enterprise Cloud data centre. The overall solution has three zones: Red, Yellow and Green. The red zone contains the input client connections. These connections may be a thin client (i.e. web browser), rich client, or mobile. The yellow zone includes an F5 load balancer to handle initial login requests and provide authentication

Figure 4.2: Enterprise Cloud Architecture Diagram



services. Requests for login are then processed using a federated identity management component. A users subscription type is stored within the BSS component. Depending on the subscription, a user will have access to one or more of the collaboration, e-mail and social components. All three data centres have the same architecture. Their only difference is geographical location.

Product development follows a CD development model whereby small component features are released to the public on a monthly basis. For each outage event, we have access to the full outage report, but we particularly focus on the time duration (recorded in minutes) between outage events (inter-arrival time). We also consider the inter-arrival time when grouped against the component, outage type and data centre.

Our third case study aims to answer the following questions: What distribution is best suited to model the inter-arrival time of Cloud outage events recorded in our dataset? Second, does the inter-arrival time distribution vary by component? Third, does the inter-arrival time distribution differ by failure category? Fourth, does the inter-arrival time distribution differ by data centre? To answer these four questions, this study is broken down into the following attributes: outage distribution, outage component, outage failure

category and data centre location.

4.2.1 Outage inter-arrival time distribution

Probability distributions are used in statistics to assign a likelihood of an event taking place. In the case of Cloud outage events, by analysing the distribution of all outages, it may be possible to fit a known distribution to our dataset. If a distribution can be fitted, these distribution properties can be used to infer the most likely outcome of an outage event. For example, a probability distribution could be used to infer the likelihood duration between outage events. Additionally, if a suitable distribution can be found, such a distribution may be used to determine the proportion of outage events less than, greater than or between two time intervals. (e.g. What proportion of outage events have an inter-arrival time greater than 100 minutes?)

An outage distribution is plotted for the complete set of outages. To determine whether our dataset can be modelled by a known distribution type, a parametric approach is considered (i.e. MLE [158]). Using the R package fitdistrplus [159] we fitted seven known distributions (e.g. exponential, gamma, log-normal and Weibull), against our dataset. For distribution validation, we used the R library ADGofTest [160] against these seven distributions types.

4.2.2 Outage component

Recognising component failures gives an understanding of a) what components are more likely to contribute to an outage event and b) the relative duration between each event concerning a component. For example, operations teams may have various probes to determine if an event is likely to cause a failure. Development and test teams may have a suite of test cases to find a certain class of issue. Outage events can provide operations teams with an understanding of potential gaps in their probes and monitoring solutions. Furthermore, for development and test teams outage events can highlight gaps in test coverage, irrespective of whether this gap is at unit, functional, or performance/system level. There are four main components in our data set: BSS, collaboration, e-mail and social. For this case study we categorised our software components as follows: BSS/social, collaboration, e-mail and mixed (where multiple components were involved).

4.2.3 Outage type

Grouping Cloud outages by type is a useful exercise. Using these subcategories: Configuration/Manual, Contention/Concurrency, Disaster Recovery, Network and Hardware/Other, we can infer whether the duration time between each type outage is similar.

Configuration/Manual errors involve situations where a configuration change is made from one value to another that causes a software or hardware component to behave abnormally. For example, a Load Balancer setting could be changed manually which reduces the throughput from Gigabits to Megabits which could significantly reduce the infrastructures' ability to, manage incoming traffic.

Contention/Concurrency outages refer to a class of issue, that is triggered by normal operations on the underlying server component code. These issues are triggered due to the inability of the code to handle either concurrent or parallel usage. Software defects may include issues related to contention under load (e.g. memory leaks, high Disk I/O, CPU usage), concurrency (e.g. deadlocks) or miscellaneous race conditions.

Disaster Recovery errors typically involve scenarios where system load was required to move from one application server or database to another. In some situations, the session data may not transfer correctly and cause a failure of routine operations.

A network error relates to a class of failure outside of misconfiguration or a hardware failure within the network infrastructure. Network failures can typically present themselves as intermittent temporary network outages, high latency/packet loss conditions or congestion based on overloading of available bandwidth. As Cloud data centres contain many distributed systems, having a reliable network infrastructure is highly desirable.

A Hardware/Other failure relates to a class of problem, which causes a piece of hardware to fail. These failures relate to a malfunction within the electronic circuits or electromechanical components (disks, tapes) of a computer system. Recovery from a hardware failure requires repair or replacement of the offending part. Additionally, the error may relate to some miscellaneous type of error that is not part of the four main failure categories.

4.2.4 Outage by data centre location

Understanding the measure of outage events at a data centre level can highlight whether a specific data centre is a factor in the duration and distribution of outage events raised. There are three data centres in our dataset: data centre A (High usage), data centre B (Low usage) and data centre C (Medium usage). Understanding whether the duration of outage events are similar or different in each data centre can be a useful data point for DevOps teams.

4.2.5 Limitations of dataset

The dataset has some practical limitations, which are now discussed. While the outage event tracking system allows for a granular categorisation system, whereby outages can be mapped to a subcomponent, there are a number of outages, which due to their severe nature can affect more than one component and subsystem. The authors reviewed the functional location of each defect to ensure precision across the analysis of the dataset. In a number of limited cases outages affected a more than one component and data centre at a time. In the case of mixed component outages, summary analysis was performed. However due to the borderline number of samples, in the case of mixed data centre outages, analysis was not performed.

The outage events that form part of this study are from an enterprise Cloud system. The inter-arrival times of these outage events apply to the software domain of BSS, collaboration, e-mail and social software. While we hope these examples will be representative of social and collaboration based software, it seems unlikely they will be typical of all types of Cloud-based software.

4.2.6 Results - outage inter-arrival time distribution

Table 4.1: Summary of Anderson-Darling GoF statistics.

Distribution name	AD statistic	p-value
Pareto	0.661	0.592
Weibull	0.975	0.371
log-logistic	1.823	0.115
log-normal	3.039	0.026
exponential	3.110	0.024
gamma	6.034	9.347e-04
logistic	12.819	2.765e-06

Figure 4.3: Four goodness-of-fit plots for Weibull, log-logistic and Pareto distributions fitted to the inter-arrival times from the Cloud outage data set

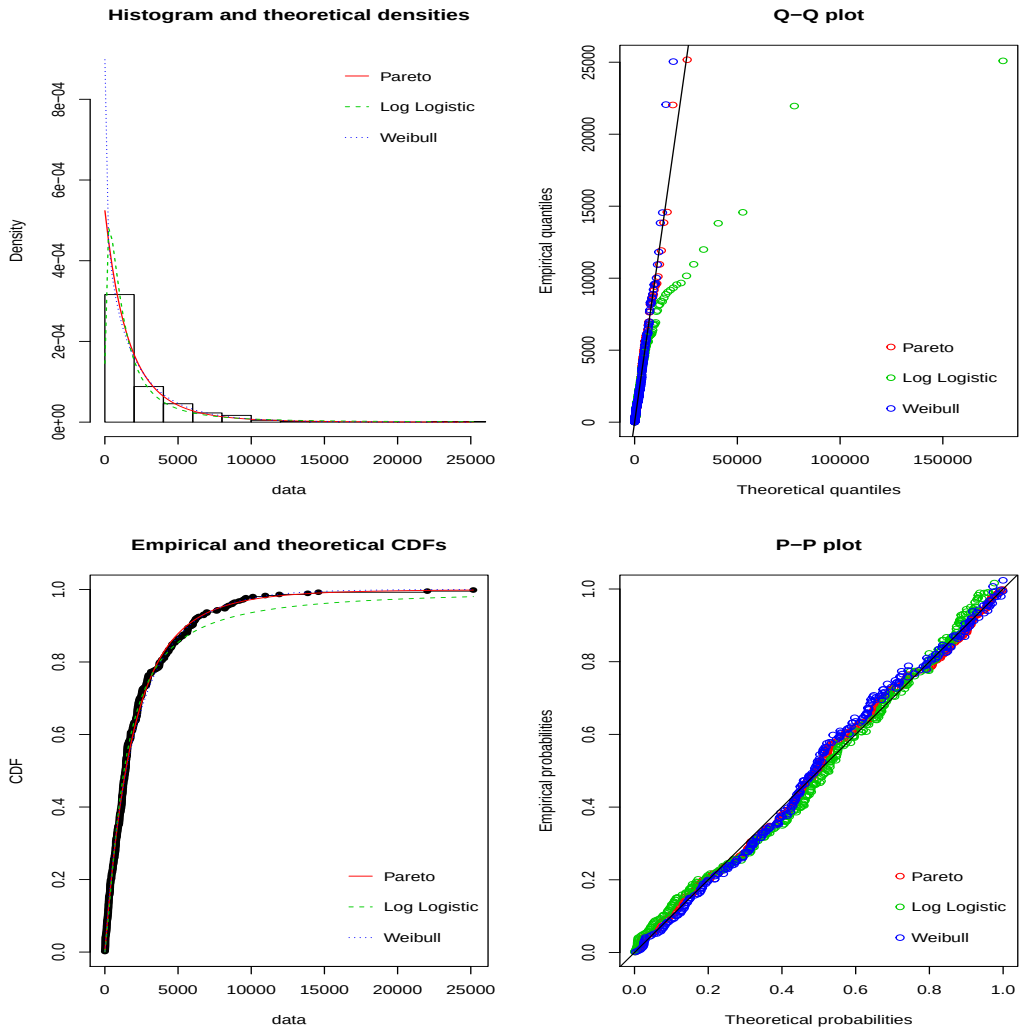


Table 4.1 shows the summary results for our distribution fitting exercise. Seven distributions types were fitted to our data set. The table is sorted from lowest to highest Anderson-Darling test statistic. Pareto, Weibull have Anderson-Darling test statistic values below 1, a value below 1 is just one indicator as to the quality of fit. Figure 4.3 shows four goodness-of-fit plots (PDF, CDF, quantile-quantile plot and probability plot) for the top three performing distributions. The quantile-quantile (Q-Q) plot contains plots of the empirical (observed) quantiles with the theoretical quantiles for each of the three distributions. A reasonable fit of the theoretical distribution to the empirical values would be indicated by this plot if the plotted values fall onto a straight

Table 4.2: Summary statistics for outage inter-arrival times by component with Pareto GoF (Mean, Std Dev and Median times are in minutes)

Statistic	BSS/Social	Collaboration	E-mail	Mixed
Samples	16	34	153	43
% Samples	7	14	62	17
Mean	2362.41	2334.26	1988.90	2567.73
Std Dev	2327.04	3106.15	2607.27	4096.10
Median	1338	1124	1262.50	1420
Skew	1.12	2.36	3.93	4.43
AD GoF (p)	0.32	0.84	4.0e-06	0.83
AD Test Statistic	1.07	0.41	Inf.	0.42

(Pareto) line sown in the Q-Q plot.

4.2.7 Results - outage component

Table 4.2 provides a summary of outage inter-arrival times grouped by component. E-Mail recorded the highest proportion of all outages with the BSS/-social category recording the lowest proportion of outages.

As a Pareto distribution was found to be the best fit for overall outage inter-arrival times, we then fitted a Pareto distribution against each outage component. Also included in the table are the AD GoF p-values and the Anderson-Darling test statistic for each sub-group of outage failures. We do this to understand if our assumption of a Pareto fit holds across all components. We note that all sub-groups have a p-value above 0.05, except e-mail. Likewise collaboration and mixed components have an Anderson-Darling test statistic of less than 0.5 while BSS/social had a test statistic of 1.07 and finally e-mail had infinite test statistic value.

4.2.8 Results - outage type

Table 4.3 provides a summary of outage inter-arrival times pooled by outage type. Configuration/Manual recorded the highest proportion of all outages while the Hardware/Other category recorded the lowest percentage of outages.

Similar to outage component, we fitted a Pareto distribution against each outage type, to test whether a Pareto distribution was present across all outage

Table 4.3: Summary statistics for outage inter-arrival times by outage type with Pareto GoF (Mean, Std Dev and Median times are in minutes)

Statistic	**Configuration Manual**	**Contention Concurrency**	**Disaster Recovery**	**Network**	**Hardware Other**
Samples	74	64	36	49	23
% Samples	30	26	15	20	9
Mean	1782.57	2342.88	2719.71	1945.13	2510.86
Std Dev	2488	2336.87	4316.33	2325.02	2346.90
Median	1019	1429	1438	1087.50	1704
Skew	5.85	1.41	3.41	1.89	0.91
AD GoF (p)	0.42	0.81	0.75	1.25e-05	0.54
AD Test Statistic	0.88	0.44	0.50	Inf.	0.73

Table 4.4: Summary statistics for outage inter-arrival times by data centre with Pareto GoF (Mean, Std Dev and Median times are in minutes)

Statistic	**Data centre A**	**Data centre B**	**Data centre C**
Samples	160	24	54
% Samples	65	10	21
Mean	2439.74	2280.21	1348.12
Std Dev	3315.80	2920.74	1500.80
Median	1395	1480.5	909
Skew	3.76	3.48	1.84
AD GoF (p)	0.54	0.79	1.18e-05
AD Test Statistic	0.73	0.45	Inf.

types. Also provided in the table are the AD GoF p-values and the Anderson-Darling test statistic for each sub-group of outage failures by type. We note that all sub-groups have an Anderson-Darling test statistic lower than 1 and a p-value above 0.05, except network.

4.2.9 Results - data centre location

Table 4.4 shows a summary of outage inter-arrival times pooled by data centre. Data centre A recorded the highest proportion of all outages while data centre

B category recorded the lowest proportion of outages. The proportion of outages follows our intuition that data centres with the highest usage have the highest proportion of failures.

Finally, we fitted a Pareto distribution against each outage by data centre, to check whether outages by data centre could be modelled by a Pareto distribution. We present the AD GoF p-values for each sub-group of outage failures. We note that all sub-groups have an Anderson-Darling test statistic value of less than 1 and a p-value above 0.05, except data centre C.

Eight outages were found in all three data centres. Due to the small number of samples detailed analysis was not performed. Furthermore, we felt it was inappropriate to merge these eight samples into one of the existing data centre pools as this may confound analysis and results from a single data centre category.

4.2.10 Discussion - outage inter-arrival time distribution

Table 4.1 shows that Pareto has the lowest Anderson-Darling test statistic and highest p-value for the Anderson-Darling test, followed by Weibull then log-logistic. Ideally the smaller the Anderson-Darling test statistic, the better the fit. If we also consider the hypothesis, do the inter-arrival times belong to a specific distribution family? In four cases we reject the hypothesis for log-normal, gamma, exponential and logistic, due to the p-value being less than 0.05. We also note that these four distributions have an Anderson-Darling test statistic of greater than 3, indicating a less than ideal fit.

Using graphical means as the second frame of reference, figure 4.3 shows the goodness of fit plots for the three best-fitted distributions. In particular, the Q-Q plot shows the plotted empirical and theoretical quantiles along with a fitted Pareto line. We observe that the plotted Pareto points fit the line in all cases apart from the second highest empirical quantile.

In the histogram plot with a fitted curve, we can see that the Pareto curve fits the curve at lower values better than the other two distributions. With a better fit at both the head and the tail of the data, the Pareto distribution is the most appropiate choice to model the inter-arrival times of our Cloud outage

data. Combining the result of the Anderson-Darling test statistic and the Q-Q plot, we note that the Pareto distribution is the best fitting distribution.

4.2.11 Discussion - outage component

Table 4.2 provides summary details of outages by component. It was observed that mixed components had the highest mean inter-arrival time with 2567.73 minutes, followed by BSS/social, Collaboration with 2362.41 & 2334.26 respectively and e-mail with 1988.90 minutes. Intuitively mixed components also had the highest median, skew and standard deviation. In three of the component sub-classes (BSS/Social, Collaboration and mixed) the fit to a Pareto distribution was above the 0.05 p-value criterion. This indicates that it is reasonable to assume the inter-arrival times from these three sub-categories can be modelled by a Pareto distribution. However, for the e-mail sub-class, the p-value computed was well below the 0.05 and an Anderson-Darling test statistic of infinite. We conclude that the Pareto distribution is not a reasonable choice to model outage inter-arrival times for e-mail alone.

Based on these results, we observe that the e-mail component has the highest proportion of outages. Moreover, the e-mail component has the lowest mean inter-arrival time. However, we note that the e-mail component recorded the second highest inter-arrival time duration (22030 minutes or 15.30 days). DevOps teams should review the root causes of each e-mail failure to determine every failure trigger. This will provide more information to understand why e-mail outages happen more frequently and with a shorter duration between failures than any other component.

Secondly, the results show that mixed components have the highest mean outage time. Additionally, we note that mixed components recorded the highest inter-arrival time duration (25172 minutes or 17.53 days). Given the complex nature of multi-component outages, it is logical to suggest that while these types of were the second most frequent, these types of outage take longer to manifest. This is due in part to the number of systems across the Cloud infrastructure that would need to fail. DevOps teams can triangulate these types of mixed outage failure across type and data centre to determine persistent failure patterns.

Finally, while the Pareto distribution is shown to be a reasonable distribution

to model outage inter-arrival times of our entire dataset, we note that when working with sub-categories, this intuition is not exclusively true.

4.2.12 Discussion - outage type

Examining outages by type gives a deeper understanding of how the inter-arrival times of outages vary by type. Table 4.3 provides summary results.

Of immediate interest is the Configuration/Manual outage type. This category scored the highest number of outages; the shortest mean inter-arrival time and the lowest inter-arrival median duration. We observe that outages related to Configuration/Manual occurred most frequently during the twelve-month analysis window. It is worth noting that the Configuration/Manual skew was the highest indicating considerable variability between consecutive outage events. Finally, we observed that the Configuration/Manual inter-arrival times alone were a good fit for a Pareto distribution with a p-value of 0.42 and Anderson-Darling test statistic of 0.88.

Contention/Concurrency and Network types recorded the 2nd and 3rd highest number of outages. The mean inter-arrival time for Contention/Concurrency was approximately 400 seconds longer than that of Network. This indicates that while Contention/Concurrency outage events happen with a greater degree of regularity than Network issues, on average the duration between subsequent events is much greater. Finally, we note that Contention/Concurrency had a highest computed p-value fit to a Pareto distribution with 0.81 and a test statistic of 0.44. However, it was also noted that the p-value of Network was below the 0.05 significance level with a infinite Anderson-Darling test statistic. This indicates that when in isolation the inter-arrival times for Network outages are not a good fit for a Pareto distribution.

Issues related to Configuration/Manual contribute most to the overall number of outages but also have the shortest inter-arrival times. Due to the complex nature of Cloud architecture, a gatekeeper style system is desirable. Such a system could self-check existing configuration setting to ensure a level of validity. Secondly, such a system could aid in the detection of configuration updates within a non-valid range being input.

With any distributed system the network health is key to infrastructure stability. The network issues studied fell into two main categories: network con-

gestion and temporary network outages. For congestion issues, business and operations teams could consider a flexible bandwidth management policy. As we shall see in our next case study, congestion issues can manifest themselves as part of misconfiguration. Therefore a holistic rather than an isolationist approach to problem determination is vital. Finally, it should be accepted that there are times when the underlying network infrastructure may fail due to unforeseen circumstances. Ensuring that an application fails gracefully can mitigate against cascade type failures.

4.2.13 Discussion - data centre location

Table 4.4 provides summary details of outages by data centre. As mentioned in our case study outline, usage varies by data centre (i.e. data centre A (High usage), data centre B (Low usage) and data centre C (Medium usage).

Data centre A recorded the highest number of outages with 160 (65% of the total). Interestingly the mean inter-arrival time was 2439.74 minutes, which is the longest mean inter-arrival time duration of all three data centres. Data centre B had the lowest number of outages (24) with the second highest mean inter-arrival time of 2280.21 minutes. Data centre C had a 2nd highest recorded outages (54) with the shortest mean inter-arrival time of 1348.12 minutes. We note that both data centres A & B had an Anderson-Darling test statistic value below 1, while data centre C had a test statistic value of infinite. We conclude that the Pareto distribution is a reasonable choice to model the inter-arrival times of outages from data centres A & B, and a poor choice to model the inter-arrival times of data centre C.

In some respects the above results are intuitive: the busiest data centre would have the most outages (given the level of concurrent users), and the least busiest data centre has the lowest number of recorded outages. However, the results of this case study illustrate that the busiest data centre has the longest mean inter-arrival time between outages. Additionally that the mean inter-arrival time between data centres does vary to a degree. We noted that that Pareto distribution was a poor fit to model the inter-arrival times of data centre C. Looking at the measures of location in Table 4.4 we can see that the inter-arrival times for data centre C are the least skewed and that the mean, standard deviation and median are within approximately 600 minutes of each

other. This indicates that the inter-arrival times of data centre C are the least heavy-tailed of the three data centres measured.

In the context of continuous delivery, new software updates are replicated to each data centre in parallel. This case study suggests that there is variability as to how end users are impacted regarding frequency and time between outage events, depending on what data centre is being accessed. With these findings, DevOps can investigate the following: Does application usage patterns vary from data centre to data centre? Additionally does data centre architecture/-configuration vary across data centres? Addressing heterogeneous infrastructure/configuration concerns can ensure problem determination is carried out across like-for-like systems.

4.3 Case study 4 - Outage service time modelling

Cloud outage studies have been shown to provide an effective way to highlight the different types of failure events [21, 22, 23, 24, 25]. For example a Outage may be caused by a software defect (Microsoft), network failure (Apple Microsoft), hardware (Salesforce) or a manual configuration change (Amazon). These studies can be leveraged by enterprises to pre-empt common failure patterns

Our fourth case study examines the same dataset as our third case study: 246 field outage events from a large Cloud-based system. The data was collected over a 12-month period (Jan 2015 – Dec 2015) and is comprised of four main components: e-mail, collaboration, social and BSS. Additionally, the type of failure events has been categorised into the following main categories: Configuration/Manual Process, Contention/Concurrency, Disaster Recovery, Network and Hardware/Other. The systems have been deployed within three data centres and are used by customers globally. The software is developed in Java and runs on Linux.

Product development follows a CD model whereby small amounts of functionality are released to the public on a monthly basis. For each outage event, we have access to the full outage report, but we particularly focus on the time (recorded in minutes) taken to resolve the outage with additional focus on the

software component and the type of error, which was the root cause of the outage. The following terminology will now be defined to provide context.

This case study aims to answer four questions. First, How are the service times of Cloud outage events distributed? Second, does the service time distribution vary by component? Third, does the service time distribution differ by failure category? Fourth, does the service time distribution differ by data centre? To answer these four questions, this study is broken down into the following attributes: outage distribution, outage component, outage failure category and data centre location.

When measuring the service time distribution either for all outages and then by component, type and data centre, we have similar considerations to those raised as part of outage inter-arrival time modelling in sections 4.2.1 through 4.2.4. Therefore we have decided not to repeat the same concerns in this case study. Additionally, the same data set limitations that were discussed in section 4.2.5 apply equally to the service time distribution modelling.

4.3.1 Results - outage service time distribution

Table 4.5: Summary of Anderson-Darling GoF statistics for service time distribution fitting.

Distribution name	AD statistic	p-value
log-normal	0.343	0.903
log-logistic	0.737	0.529
Pareto	1.602	0.154
exponential	3.110	0.024
Weibull	6.828	4.e-04
gamma	272.44	1.796e-06
logistic	Inf	1.796e-06

Table 4.5 provides a summary results for our distribution fitting exercise. Seven distributions types were fitted to our data set. The table is sorted from lowest to highest Anderson-Darling test statistic. log-normal and log-logistic and have test statistic values below 1, which is one indicator for goodness-of-fit. Figure 4.4 shows four goodness-of-fit plots (PDF, CDF, quantile-quantile plot and probability plot) for the top three performing distributions. A log-normal distribution was chosen as it had the lowest AD test statistic, the majority of points fitted the line within the Q-Q plot and highest p-value.

Figure 4.4: Four goodness-of-fit plots for log-normal, log-logistic and Pareto distributions fitted to the service times from the Cloud outage data set

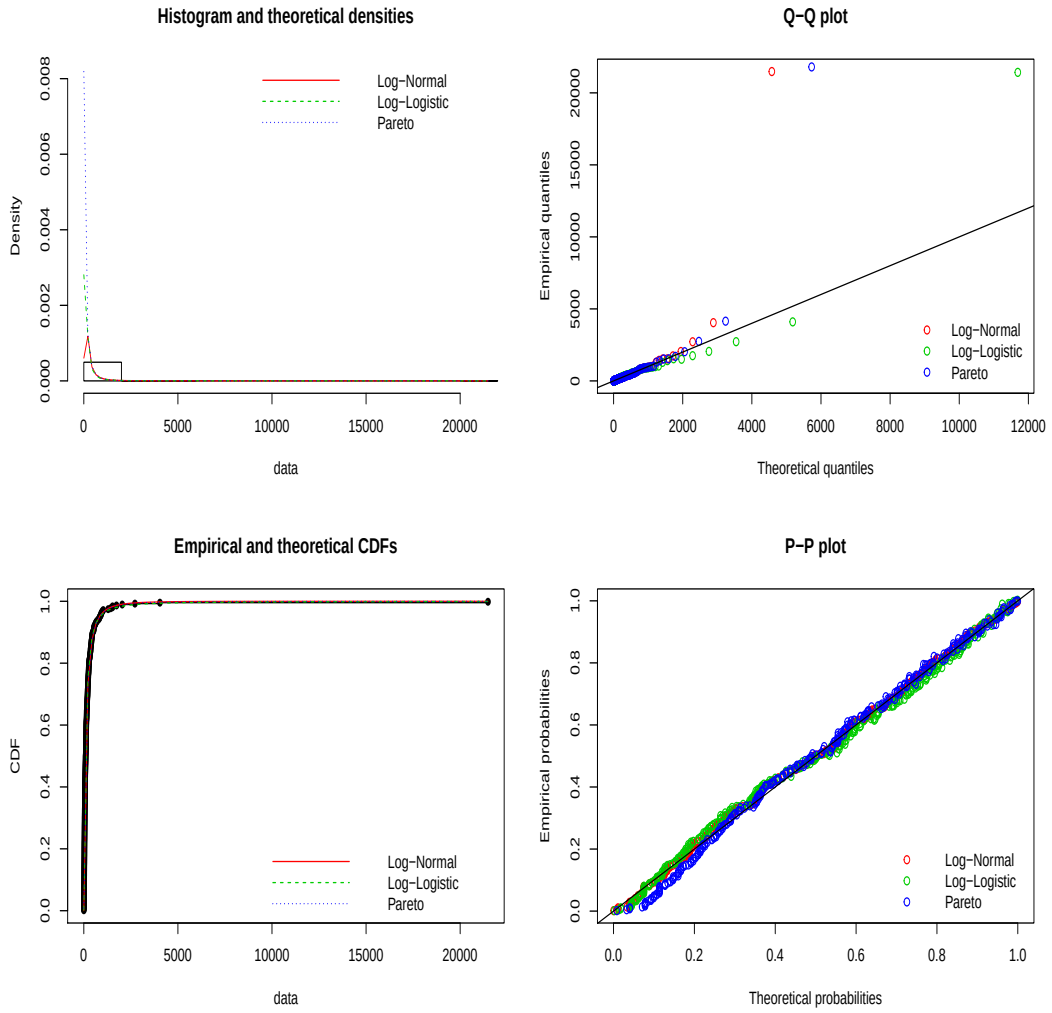


Table 4.6: Percentiles and computed values for the log-normal distribution at six percentile locations

Percentile	Xp	Value
1st	X0.01	4.77
10th	X0.10	18.49
50th	X0.50	18.49
90th	X0.90	513.47
99th	X0.99	1990.70

Table 4.7: Summary statistics for outage service times by component with log-normal GoF (Mean, Std Dev and Median times are in minutes)

Statistic	BSS/Social	Collaboration	E-mail	Mixed
# Observations	16	34	153	43
% Observations	7	14	62	17
Mean	274	189	258	626
Std Dev	639	379	423	3261
Median	45	61.5	126.5	85
Skew	3.56	3.83	5.45	6.30
AD GoF (p)	0.69	0.62	0.99	0.64
AD Test Statistic	0.55	0.61	0.20	0.61

Table 4.6 provides a summary of the computed values of x at six percentile locations.

4.3.2 Results - outage component

Table 4.7 lists the summary statistics of outage events broken down by component. E-Mail recorded the highest proportion of all outages. BSS/social recorded had the lowest. Outages are most likely to happen in the e-mail component. We note that mixed component outages require the longest time to resolve. Additionally, we indicate that a log-normal fit is a good fit irrespective of the component, given the test statistic value of below 1.

Due to the small number of samples (16) recorded for the BSS/social category, the goodness of fit value should be treated with caution.

4.3.3 Results - outage type

Table 4.8 lists the summary statistics of outage events broken down by type. Configuration/Manual and Contention/Concurrency recorded the highest proportion of outages while Hardware/Other had the lowest. Outages are most likely to be either Configuration/Manual or Contention/Concurrency. Configuration/Manual outages take the longest time to resolve, while disaster recovery outages take the shortest time to resolve. Furthermore, we note that

Table 4.8: Summary statistics for outage service times by type with log-normal GoF (Mean, Std Dev and Median times are in minutes)

Statistic	**Configuration Manual**	**Contention Concurrency**	**Disaster Recovery**	**Network**	**Hardware Other**
# Observations	74	64	36	49	23
% Observations	30	26	15	20	9
Mean	488	239	134	315	243
Std Dev	2488	469	161	591	358
Median	114.5	86	72	145	91
Skew	8.28	3.69	2.33	5.30	2.11
AD GoF (p)	0.91	0.97	0.94	0.75	0.96
AD Test Statistic	0.34	0.25	0.30	0.50	0.27

the skew and standard deviation is highest for configuration/manual outages, indicating high variability in the time to remediate such events. Finally, we note that a log-normal distribution is a reasonable fit regardless of outage type.

Due to the small number of samples (23) for the Hardware/Other category, the goodness of fit value should be treated with caution.

4.3.4 Results - Data centre location

Table 4.9 lists the summary statistics of outage events broken down by data centre. Data centre A recorded the highest proportion of outages, while Data centre B had the lowest. The remaining 3% were from outages found in all three data centres. Outages are most likely to happen within Data centre A. Data Centre C had the highest mean service time, standard deviation and skew. Data centre B had the lowest mean service time, standard deviation and skew. We note that the Anderson-Darling test statistic is 0.20 for data centre A and B and 1.1 for data centre C. These values indicate a reasonable fit irrespective of data centre.

Table 4.9: Summary statistics for outage service times by data centre with log-normal GoF (Mean, Std Dev and Median times are in minutes)

Statistic	Data centre A	Data centre B	Data centre C
# Observations	160	24	54
% Observations	65	10	22
Mean	224	188	645
Std Dev	313	280	2961
Median	113.5	89.5	79.5
Skew	2.93	2.89	6.67
AD GoF (p)	0.99	0.99	0.31
AD Test Statistic	0.20	0.20	1.1

Eight outages were found in all three data centres. Due to the small number of samples detailed analysis was not performed. Furthermore, we felt it was inappropriate to merge these eight samples into one of the existing data centre pools as this may confound analysis and results from a single data centre category.

4.3.5 Discussion - outage service time distribution

To answer the question how are the service times of Cloud outage events distributed, we conducted an Anderson Darling goodness of fit test against seven distributions; Exponential, Gamma, log-normal, log-logistic, logistic, Pareto and Weibull. Table 4.5 provides a summary of the computed Anderson-darling test statistic and p-values for seven distributions fitted.

In four cases we note that the Anderson-darling test statistic for exponential, Weibull, gamma and logistic distributions are all above 3 indicating a poor fit. From the perspective of a hypothesis test, is there evidence to suggest the observed data may be drawn from either of these four distributions in all cases as the computed p-value is below 0.05, this hypothesis is rejected.

Table 4.5 also demonstrates that it would be reasonable to assume our dataset could be drawn from either a log-normal, log-logistic or Pareto distribution. However, given the very low AD test statistic and high p-value indicates that a log-normal distribution is the best fit for our data.

Using figure 4.4 as a means of graphical inspection, we can see that all three

distributions provide a reasonable fit. Looking at the Q-Q plot, we note that the empirical and theoretical quantiles are plotted against a theoretical log-normal line. We note that two quantile points from the log-normal distribution do not pass through the line. The P-P plot also indicates a good fit for all three distributions. However the Pareto distribution does not appear to fit as well for lower probabilities unlike log-logistic and log-normal, this may account for the smaller p-value.

While the computed Anderson-darling test statistic and p-values suggest that log-normal is a reasonable fit for our dataset, the Q-Q plot indicates that our data is a fair fit except for one extreme value. As a consequence, the log-normal distribution may be suitable for general service times, but for extreme service times, the distribution may be unsuitable.

Our findings further expands the applicability of the use of the log-normal distribution of model repair times. Prior literature suggests that repairable systems typically refer to mechanical, electric and electronic systems [161]. However, given the above results, we can now include software systems as another subtype.

Given the nature of Cloud computing, new code updates and configuration changes are made on a regular basis. It is not uncommon for an enterprise to introduce changes on a bi-weekly or monthly basis. Therefore with this high level of system activity, it is surprising that outages can occur frequently. If a state of the art outage tracking system were introduced, it would be interesting to determine overall as both process improvements were made coupled with underlying code stability to observe the overall effect on both the distribution type and shape. This would provide a concrete answer to questions such as: what impact do specific process improvements make to overall outage times? As a business where do resources need to be deployed to improve platform stability: Development, Operations or Quality Assurance?

4.3.6 Discussion - outage component

Examining outages by component can give insight as to which component is likely to exhibit outages and whether these times vary by component.

Table 4.7 provides summary details of outages by component. It was noted that mixed components had the highest mean outage time of 627 minutes,

followed by BSS/social, e-mail with 274 & 258 minutes respectively and collaboration with 189 minutes. Consequently, mixed components also have the highest standard deviation and skew. In all cases, each component class had a reasonable fit to a log-normal distribution, with the e-mail category fitting best with a p-value of 0.99 and an Anderson-Darling test statistic of 0.20. However, the BBS/Social category has a low number of samples. Therefore the goodness of fit assessment of this category should be treated with some caution.

Based on these results, we note that the e-mail component has the highest proportion of outages. Tiger teams (i.e. a cohort of e-mail developers and DevOps) should review the root cause of each outage related to the e-mail component, as these experts are best positioned to expedite root cause analysis. This will gain understanding as to the what types of failures contribute most to e-mail outage events. Triangulating each outage event against the failure type and data centre location can help business and operations teams resource their crisis teams on a per-component basis.

Secondly, the results show that mixed components have the highest mean outage time. This result seems logical, given that when an outage occurs across common infrastructure and multiple components that the repair time is greater. There are many systems to check and repair as part of the remediation process. To verify, the individual reports were checked for mixed component failures. It was found that one outage took multiple days to resolve. Hence skewing the overall mean time. While this data point may be considered an outlier in the classic sense, given this was a real fault, it must be included as part of an analysis. Tiger teams should determine the root cause of each outage to intersect failure type and data centre to understand common failure patterns.

4.3.7 Discussion - outage type

Examining outages by type gives a deeper understanding of what types of problems are likely to cause an outage within a Cloud infrastructure. Table 4.8 provides this insight.

Significantly Configuration/Manual had the highest expected outage time with 489 minutes, with Network next highest with 315 minutes. Contention/Con-

currency, Hardware and Disaster recovery had expected outages times of 239, 243 and 134 minutes respectively. Finally, the outage times of each category were fitted with a log-normal distribution. In each case, the hypothesis of whether a log-normal distribution was a suitable distribution could not be rejected, given the low an Anderson-Darling test statistic of values computed on a per component basis. However, one caveat is that the Hardware/Other category had a low number of samples, so this result must be treated with caution.

Speaking generally, the more observations provided the better the level of precision in terms of an overall result (i.e. whether a hypothesis is accepted or rejected). If only a small number of observations are available, looking at the observations as a numerical sample rather than part of a sample of wider distribution should be considered. If a sufficient level of observations are available and an appropriate distribution can be fitted, we can calculate the percentiles at distinct values to determine a range of values between two distinct percentile points as can be seen in table 4.6

We note that issues related to Configuration/Manual contribute most to the overall number of outages but also take the longest to resolve. Given the relative complexity of the entire Cloud architecture, it is apparent that a system of managed configuration changes is required. Firstly to ensure that for all configuration changes made, that there is a commit and rollback feature to ensure that harmful (extreme) configuration settings can be reversed if required. Additionally, tiger teams should implement a system, which can monitor real-time configuration changes across all data centres.

With any distributed system the network health plays a significant role in system stability. The network issues studied fell into two main categories: network congestion and temporary network outages. For congestion issues, business and operations teams need to define clear bandwidth capacity requirements to ensure that their infrastructure has the bandwidth to meet the demands of their existing user base and future subscription signings. The underlying application code and middleware stack should have additional resiliency added to ensure that temporary outages do not cause cascade failures.

4.3.8 Discussion - data centre location

Table 4.9 provides summary details of outages by data centre. As discussed previously in our case study outline, user concurrency varies by data centre. Data Centre C had the highest mean outage time of 645 minutes, while data centre's A & B had mean outages times of 224 and 188 respectively. All three data centre outage times were modelled with a log-normal distribution and both data centre A & B were a reasonable fit. Each had a p-value of 0.99 and an Anderson-Darling test statistic of 0.20. Data centre C fared worst regarding fitting with a p-value of 0.31 and an Anderson-Darling test statistic of 1.1.

In some ways the above results are expected, it seems intuitive that a high use data centre would incur the most outages due to the high level of customer activity, however even with all these outages the mean outage time is 224 minutes, which is approximately 90 minutes less than the overall mean. What appears somewhat counterintuitive is that data centre C has the second highest number of outages and the highest mean outage time. From closer inspection the mean outage time of data centre C is due to a small number of outages with high duration. Finally, it is worth noting that data centre B has the lowest number of outage events and the lowest mean outage time.

In the context of software delivery to multiple data centres, the same code is released to each system. Customers are impacted in different ways depending on which data centre is used. With this knowledge, tiger teams can investigate in two areas. Firstly does the underlying customer use case of each data centre vary? Secondly, a root and branch investigation of each data centre configuration should be conducted and compared for discrepancies, with a specific focus on the configuration of the e-mail component.

4.4 Conclusion

The purpose of our case studies conducted in this chapter was to model the inter-arrival time and service time duration of a data set of Cloud outage events. Our key aim was to understand if either or both durations could be modelled by a known probability distribution. We found that the Pareto distribution is a useful distribution for modelling the inter-arrival times of Cloud outages. We also found that the log-normal distribution is a useful

distribution for modelling repair times of SaaS outages. The findings of this study support previous research particularly in the field of system reliability and repair times.

Previous studies [33, 34, 35, 36, 37, 38] have shown that Cloud outages are an infrequent occurrence. Additionally, we show that the log-normal distribution is a useful tool for modelling repair times in mechanical and electronic maintainable systems.

This work adds to the existing literature in the area of modelling inter-arrival and service times. Our contributions provide additional results as to how repair times can vary between failure type, component and the data centre used at the time of a Cloud outage.

For enterprises similar to which the data set was gathered, the key points as follows. Modelling both the inter-arrival and services times is useful, as it can help determine an underlying distribution. This distribution can be used to calculate a proportion of inter-arrival or service times between a given time interval. Additionally, modelling by component, outage type and data centre can also be useful to determine what proportion of outages lie between a particular range. This more in-depth analysis may also be helpful to determine whether a portion of inter-arrival or service times are aligned to a distinct component outage type or data centre.

For the SME and small teams, the use of described modelling techniques may also be transferable to a degree. Given their small pool of resources the result of inter-arrival and service time proportions may generalise and help with resource allocation. For example, being able to determine the proportion outage service times that exceed a specific duration may be of value in terms of resource deployment, however additional investigation would be required to determine how useful the modelling techniques could ultimately be to a small team.

In our next chapter, we shall assess the results of our inter-arrival and service models to understand how Cloud failures can be simulated over a period of time. Additionally, we consider how DevOps can be deployed using this simulation model.

CHAPTER 5

Outage Simulation

As SMEs adopt Cloud technologies and rapid delivery models to provide high-value customer offerings, there is a clear focus on uptime. Cloud outages represent a challenge to SMEs and micro teams to maintain a services platform. If a Cloud platform suffers from downtime, this can have a negative effect on business revenue. Additionally, outages can divert resources from product development/delivery tasks to reactive remediation. These challenges are more immediate for SMEs or micro teams with a small pool of resources at their disposal. In this chapter, we present a framework that can be used to model the arrival of Cloud outage events. Such a framework can be used by Cloud operations teams to manage their scarce pool of resources to resolve outages, thereby minimising the impact on service delivery. We analysed over 300 Cloud outage events from an enterprise data set. We modelled the inter-arrival and service times of all outage events and found a Pareto and log-normal distribution to be a suitable fit. We used this result to produce a special case of the G/G/1 queue system to predict the busy times of DevOps personnel. We also investigated dependence between overlapping outage events. Our busy time predictive queuing model compared favourably with observed data, using one million simulations.

5.1 Introduction

Cloud outage prediction and resolution is an important activity in the management of Cloud services. Recent media reports have documented cases of

Cloud outages from high profile Cloud service providers [25]. During 2016 alone the CRN website documented ten high profile Cloud outages (See table 2.1 for more details). Due to the increasingly complex nature of the data centre infrastructure, coupled with the rapid continuous delivery of incremental software updates, it seems that Cloud outages are with us for the time being.

For operations teams that maintain a Cloud infrastructure, they rely on state of the art monitoring and alert systems to determine when an outage occurs. Examples of monitoring solutions include: New Relic [162], IBM Predictive Insights [163] and Ruxit [164]. Once a new outage is observed, depending on the outage type (e.g. Software component, infrastructure, hardware etc.), additional relevant experts may be called to remediate the issue. The time taken to resolve the issue may depend on a number of factors: the ability to find the relevant expert, swift problem diagnosis and velocity of pushing a fix to production systems.

Both SMEs and micro teams within large organisations face a number of challenges when adopting a Cloud platform and a mechanism to deliver products and services. A number of recent studies have outlined that both frequency and duration of outage events are key challenges [21]. Almost all European SMEs (93%) employ less than ten people [157]. Ensuring that adequate skills and resources are available to accommodate incoming outage events is highly desirable.

In this chapter, we propose a framework that micro teams or SMEs can leverage to best manage their existing resource pool.

The core idea of this framework is for operations teams to use a special case of the G/G/1 queue to model the inter-arrival and service times of outage events. This chapter consists of a case study of outage event data from a large enterprise dataset. By modelling both inter-arrival and service outage times, a special case of the G/G/1 queue is developed. This G/G/1 queue is then tested against an off the shelf queue model (M/M/1) to compare and contrast queue busy time prediction. Finally, our framework also considers dependence between overlapping outage events.

To help researchers reproduce and extend this study, the source-code (written

in C) of the queue modelling framework is provided [1]. By utilising this queue framework, researchers will have the ability to test their preferred case of the G/G/1 model against the M/M/1 model.

It should be noted that the branch of probability theory called Renewal Reward Process [165][166] was also considered as part of this study. In fact, on closer inspection, the processes are quite similar. For example, the time between successive outages on a Cloud system could be assigned to the holding times; the rewards may be seen as the repair costs (service times) for each outage event. However, a key application of this study is to understand whether a queue model could be used to simulate outage arrivals. If an effective simulation model can be found, can we use this model to aid resource planning of personnel within a micro team or SME?

By looking at the traffic intensity and the queue length, we can determine how resources need to be deployed within a micro team or SME. The applications of a Renewal Reward process relate more typically to the understanding of the age of a software component, the remaining lifetime of a software component, and the replacement reward/penalty of a software component. These questions are all valid; we discuss these applications briefly in our conclusion.

Downtime is bad for business. Whether a company provides a hosting platform, more commonly known as Platform as a Service (PaaS), or for a company that consumes such a platform to deliver their services, more commonly known as Software as a Service (SaaS). The result is the same: Business disruption lost revenue and recovery/remediation costs. A recent US study looked at the cost of data centre downtimes and calculated the mean cost to be $5617 per minute of downtime [167].

In the current literature, a framework to model Cloud outage events is absent. This study observed that outage events arrive over a period of time, which require fixing to return a system to a steady state. With these attributes in mind, our literature search focuses on queuing theory and distribution fitting for repairable systems.

Another consideration is the idea of event dependence. Typical off the shelf single-server queue models such as M/M/1 and G/G/1 assume that the inter-

[1] https://github.com/flop71/borogoves

arrival and service times between events are independent [9]. However, if some form of dependence is found between events, how useful would a queuing model which assumes independence compare against that of a queuing model with dependence properties.

This study aims to answer a number of questions. First, how are the inter-arrival times of Cloud outage events distributed? Second, how are the service times of Cloud outage events distributed? Third, how can an effective queuing model be built to simulate outage event traffic? Fourth, how are inter-arrival and service times correlated? Fifth, are overlapping outage events related or can we treat each event as independent?

Further motivation is driven by recent reports and studies into the adoption of Cloud computing. Carcary et al. [32], Alshamaila et al. [168] and Oliveira et al. [169] all conducted studies on Cloud computing adoption by SMEs. The consensus is that there is no single factor which impedes Cloud adoption. The main constraints noted were: Security/compliance/data protection along with geo-restriction and compatibility with legacy systems. It was also noted that manufacturing and services sectors have different concerns concerning Cloud computing adoption. Which indicates some level of customisation is required to meet the needs of different industries. Additionally, Gholami et al. [170] provided a detailed review of current Cloud migration processes. One of the main migration concerns mentioned was the unpredictability of a Cloud environment. Factors that led to this unpredictability included: Network outages and middleware failures. The study concluded that a fixed migration approach is not possible to cover all migration scenarios due to architecture heterogeneity.

As can be seen from the literature review in chapter 2, a number of studies have been conducted into Cloud outage failures and the inter-arrival / repair times of computer systems. However, there are no studies that conduct end-to-end research of outage events to build a queue model to predict the likely busy time and resource management of DevOps teams. Our proposed queue model aims to plug the gap that has been identified in the current literature.

Table 5.1: Summary of dataset metrics

Metric	Value
Number of outage events	331
Data collection duration	18 months (January 2015 to June 2016)
Software components	Business Support System (BSS), Collaboration, E-mail, and Social
Number of Data Centres	Three
Programming Language	Java
Operating System	Linux
Hardware	Intel based. (Mixture of physical and virtual systems)
Load Balancing	F5

5.2 Case study 5 - outage simulation

Cloud outage studies have been shown to provide an effective way to highlight common failure patterns [21]. In this and subsequent sections, our study will present a data set and queuing model. The aim of which is to illustrate its efficacy in modelling Cloud outage events.

Our dataset is taken from an enterprise Cloud system which details all outage events over an eighteen month period. The dataset is an expanded version of the dataset that was used in chapter 4. The main difference is that for this case study, we had six months of additional outage data available. We note that the initial analysis steps in this chapter are similar to chapter 4 (i.e. inter-arrival and service distribution modelling).

In each case, we had access to the full outage report log. For each outage, we observed the arrival time of each event and how long each outage took to repair. With each arrival time known the inter-arrival time between each outage was computed. We used the repair time duration as the service time.

A number of points related to the dataset are summarised in Table 5.1.

It is important to affirm the Programming Language, OS, Hardware and Load Balancing solution used as part of the Enterprise solution. These metrics may

be underlying factors in the root cause of the observed outages. Further investigation into the root cause of each outage event is beyond the scope of this case study. We discuss this matter in our study limitations subsection below.

Product development follows a CD model whereby small amounts of functionality are released to the public on a monthly basis. This study focuses on the following aspects of the outage event data: The inter-arrival time between each outage, the time to service each outage event and whether or not overlapping outage events are related.

Before outlining our research questions, it is useful to understand why queuing theory could be used to model Cloud outages events. Outages begin at a specific point in time. The problem is then diagnosed and serviced by tiger and DevOps teams. These characteristics are very similar to the properties of a queue system (i.e. inter-arrival times, service times and queue length).

93% of micro teams and SMEs have less than ten employees [171], yet are adopting the Cloud as a method to deliver software and services. Given the unpredictability of Cloud infrastructure architecture, this study is required to understand whether a micro team / SME has adequate resources to manage future Cloud outage events.

5.2.1 Inter-arrival time distribution

Probability distributions are used in statistics to infer how likely it is for an event to happen. In the case of Cloud outage inter-arrival times, we can analyse the data and determine which distribution is the best fit. The properties of a distribution can then be used to infer the probability of an event is happening. For distribution fitting, we used the R package fitdistrplus [159] to fit various distributions to our dataset. To validate the efficacy of each distribution, the authors used the R package ADGofTest [160] which uses the Anderson-Darling goodness of fit test to determine if the observed data follows a specific distribution [77]. We use the same distribution modelling approach discussed in chapter 4.

5.2.2 Service time distribution

This study has similar motivations for Cloud outage service times. Being able to determine a probability distribution that best fits this outage event dataset is a useful exercise. By combining both inter-arrival and service time distributions, a queue system can then be built. This queue system can be used to model the arrival and service times of Cloud outage events. The approach to distribution fitting and validation is the same as described in chapter 4.

5.2.3 Outage event simulation framework

Queuing models have been used previously across many sciences to simulate the arrival and service times for a collection of events. Typically observed inter-arrival and service times data is used to derive a suitable fitting distribution. After that, the distribution parameters (i.e. mean, rate, shape, scale etc.) are used to simulate queue traffic. Simulations allow large experiments to be undertaken which could not be conducted with the real system, and to make predictions of future behaviour.

For this study, we look at how a queuing system can be used to model Cloud outage events. Our queue system was developed using the C programming language. Our study conducted one million simulations against a G/G/1 system based on fitted sample distributions. Furthermore, our study also conducted the same number of simulations against an M/M/1 queue. For the mean inter-arrival and service times, we used the computed means from our Pareto and log-normal distributions.

An assessment of the usefulness of such simulations is given within the context of resource management within a micro team or SME. Can such simulations provide a reasonable degree of precision to aid resource planning of DevOps / Tiger teams with constrained levels of staffing?

5.2.4 Correlation between inter-arrival and service times

Statistical correlation is used to measure how two variables are related. For this study, we want to check if there is a relationship between the duration of inter-arrival and service times and if so what is the level of this relationship.

There are a number of tests that can be conducted to determine correlation. We shall discuss these briefly.

Pearson [172] and Spearman's [173] ranked coefficient is a measure of the strength of a linear association between two variables and is denoted by r. The strength of the relationship is measured between 0 (no correlation) to 1 (high correlation). Additionally, the coefficient can be positive or negative indicating the type of relationship. Pearson's test is typically used when dealing with variables with a linear relationship while Spearman's test can be used where a relationship is monotonic (whether linear or not).

Simple linear regression [87] is a statistical method that allows us to summarise and study relationships between two continuous variables. One variable, denoted x, is regarded as the predictor or independent variable. The other variable, denoted y, is regarded as the response or dependent variable.

Finally autocorrelation [174] is the correlation of a variable with a lagged version of itself. The test looks at the time lag between events to infer if a repeating pattern (seasonality) exists. Examining the lags of variables can be useful to determine if there are distinct cyclical patterns between variables or if these patterns are simply white noise.

For our correlation assessment we used the following functions found in the base R package: cor.test [175], lm [176] acf [177] to test the relationship between inter-arrival and service times.

Correlation tests can also be used to determine dependence between variables. However, we shall discuss a specific aspect of dependence in the following section.

5.2.5 Assessment for no association and linkage between overlapping outage events

As discussed earlier, the M/M/1 queuing system assumes that arrivals are independent. This is due to the understanding that both arrival and service times are governed by a Poisson process. For G/G/1 (i.e. general distribution) queues, we still have the same assumption of independence. However, the occurrence of cascading (i.e. dependent) outages events can play a role in the shape of both inter-arrival and service distributions. Therefore for the final

part of our statistical analysis, this study formally tests whether the arrival times of overlapping outage events are independent or not.

The following method will be used: Defect outage reports will be analysed to determine if an arrival overlaps with the service time of a prior outage event. Next, the outage reports will be examined to determine if the two overlapping outages are related by component area and root cause. The outage counts will then be arranged in a two by two contingency table format. Fishers exact test for independence[178][179] is then conducted. For the actual test, the authors used the R library fisher.test [180].

5.2.6 Study limitations / Threats to validity

The dataset has a number of practical limitations, which are now discussed. The event data collected for this study is comprised of outage reports from an enterprise system deployed over three data centres. For event modelling, the authors have assumed a simple queue. In other words a queuing system with one "server". Given the lack of studies in the area of modelling Cloud outage events within a queuing framework, the authors wanted to validate such a framework in the context of a simple queue initially.

Bearing in mind that an outage of a specific type (i.e. Hardware, Network, High Availability) will typically affect a single data centre at a time, there is a class of outage (i.e. Software, Configuration) which may affect multiple systems concurrently and in parallel. Our queue model could be extended to a queue with multiple servers using the following approach. The inter-arrival and service times for each data centre could be modelled separately. Using this result, we could obtain a per data centre queue busy time. For outages which affect multiple data centres, we can model these events in isolation to determine the probability of such an event occurring. Next, we can analyse within the context as an overlapping event within the entire multi-server data centre infrastructure.

For the M/M/1 simulation, in the absence of a suitably good fitting mean parameter we used the means from our inter-arrival and service time distributions.

The outage events that form part of this study are from an enterprise Cloud system. We recognise that the outage events observed can have a variety of

Table 5.2: Inter-arrival time distributions : Goodness of fit summary

Distribution	AD Test Statistic	p-value
Pareto	0.53	0.72
log-logistic	1.93	0.10
log-normal	3.79	0.01
gamma	632.89	1.83e-06
exponential	Infinity	1.83e-06
logistic	Infinity	1.83e-06
Weibull	Infinity	1.83e-06

root causes (i.e. Configuration-Manual, Concurrency-Contention, Hardware, High availability and Network). A number of Concurrency-Contention outage events may be tied to the code base which is specific to this Cloud system. However, concurrency and contention issues can occur in distributed computing systems irrespective of the underlying software code.

The hardware used in the Cloud environment is Intel-based (i.e. x86_64). Therefore any hardware failures will apply to this hardware form factor. Additionally, the programming language used to code the software services is Java. Any outage events that are related to a software defect should be compared to similar Cloud outage data where Java is used.

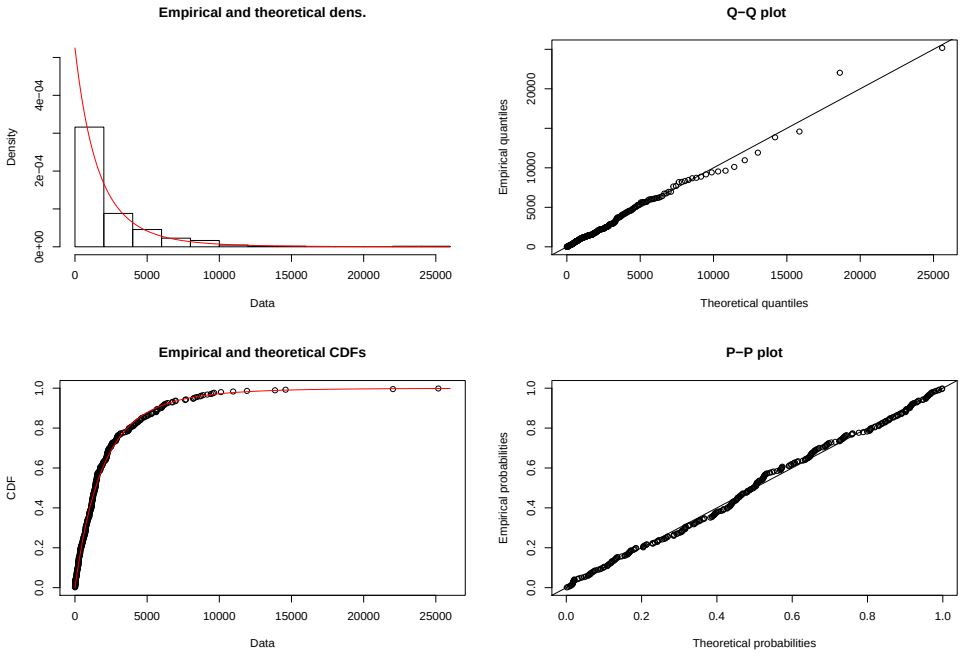
The outage events apply to the software domain of BSS, Collaboration, Email and social applications. As a consequence, the analysis may not be relevant outside of this realm.

5.2.7 Results - Inter-arrival time distribution

Table 5.2 shows a summary of the seven distributions fitted against the observed inter-arrival time data. Each distribution is listed along with its corresponding Anderson Darling test statistic and p-value. Figure 5.1 shows four Goodness-of-fit plots for a fitted Pareto distribution: Density, Cumulative Distribution Function (CDF) , Probability (P-P) [181] and Quantile (Q-Q) [182].

Our hypotheses question asks: Does the dataset follow a specified distribution? We then use a corresponding p-value to test if the data set comes from a chosen distribution. If the p-value is less than 0.05, we can reject the hypothesis (i.e.

Figure 5.1: Density, CDF, P-P and Q-Q plots for a fitted Pareto Distribution against inter-arrival time data



our data set is unlikely to follow a specific distribution). If the p-value is greater than 0.05, we cannot reject our hypothesis (i.e. our data is likely to follow a specific distribution).

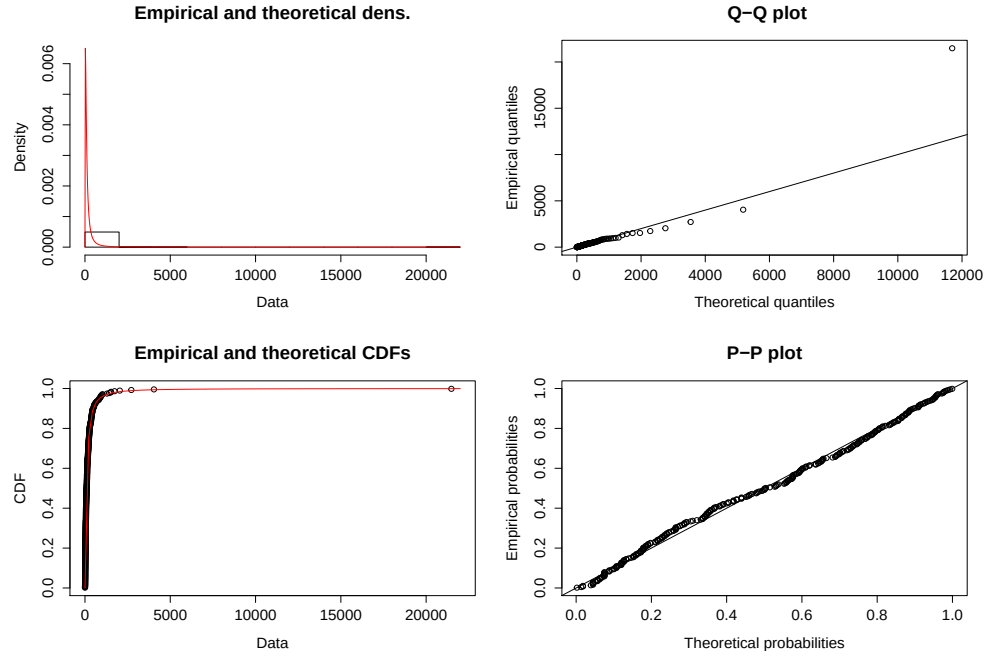
Except for Pareto and log-logistic distribution, all others were a poor fit indicated by the low p-value and the very large AD test statistic. Pareto was found to be the best fit with an AD test statistic of 0.53 and a p-value of 0.72. It is worth noting that as the AD test statistic becomes large, the corresponding p-value remains fixed, which explains why the four worst fitting distributions have identical p-values.

Figure 5.1 graphically illustrates the how well the Pareto distribution fits our dataset. The Q-Q plot shows the majority of data fits the distribution model line, except a number of large quantiles residing outside the model line. Additionally, the P-P and CDF plot also indicate that our data set is a good fit for Pareto with the majority of points positioned along the model line/curve. By and large, all points reside on the model line except the probability values between 0.37 and 0.57. However, this observation does not significantly

Table 5.3: Service time distributions : Goodness of fit summary

Distribution	AD Test Statistic	p-value
log-normal	0.34	0.90
log-logistic	0.74	0.53
Pareto	1.60	0.15
Weibull	6.82	4.00e-04
gamma	272.44	1.83e-06
exponential	Infinity	1.83e-06
logistic	Infinity	1.83e-06

Figure 5.2: Density, CDF, P-P and Q-Q plots for a fitted log-normal Distribution against service time data



undermine the assumption that the Pareto distribution is a reasonable fit for our data set.

5.2.8 Results - Service time distribution

Table 5.3 shows a summary of the seven distributions fitted against the observed service time data. Each distribution is listed along with its corresponding Anderson Darling test statistic and p-value. Figure 5.2 shows four Goodness-of-fit plots for a fitted log-normal distribution: Density, (CDF),

Table 5.4: Summary of results from queue modelling experiments and observed overlapping outage events

Model Type	% Busy	% Free
Observed Data	7.9	92.1
Simulation (G/G/1)	5.7	94.3
Simulation (M/M/1)	3.0	97.0

(P-P) and (Q-Q).

For the second research question: how are the service times of Cloud outage events distributed, again seven continuous distributions were fitted against the data set. Using the same method for inter-arrival times, an AD Goodness of fit test statistic and the p-value was computed for each distribution. We also pose the same hypothesis question: Does the dataset follow a specified distribution?

Both log-logistic and Pareto scored well, however log-normal was found to be the best fitting with an AD test statistic of 0.34 and a p-value of 0.90. All other distributions had a p-value of <= 0.15. Once again we can see that as the AD test statistic becomes large, the corresponding p-values become fixed around a value of 1.83e-06.

The plots contained in Figure 5.2 show how the log-normal distribution is a reasonable fit to our dataset. For the Q-Q plot, the majority of values fit the distribution line, however there is a clear extreme value that does not fit. That said there are a very small number of values that stray from the line, with one obvious extreme value. By and large, the fit is very good. Additionally, for the P-P plot, the values from our dataset either reside on or very close to the line which illustrates the quality of fit.

5.2.9 Results - Outage event modelling framework

Now that we have shown the results of distribution fitting, we shall now use these distributions to test our special case of our G/G/1 queue model. For the Pareto distribution, our rate and shape parameters were estimated to be 4.94 and 9404.06 respectively. Our log-normal service distribution had a computed location and scale parameter of 4.58 and 1.30 respectively.

Table 5.4 shows a summary of the queue model experiments conducted as well as details of the observed outage data over an eighteen month period. The model type defines whether observed data or a simulation was conducted. The type of simulation is also included. The % Busy and % Free columns relate to the number of overlapping events in the queue. Specifically, we counted the number of times an outage event (either observed or simulated) entered the queue system while an existing outage was currently being serviced. This value is presented as an overall percentage.

As we can see from Table 5.4, for the observed data the queue was free approximately 92% (i.e. either 0 or 1 outage was being served) and approximately 8% of the time the queue was busy (i.e. while an outage was being served another outage event arrived). Comparing the results of both simulations: the G/G/1 simulation compared favourability with the observed results with approximately 94% and 6% free and busy time. However, the M/M/1 simulation compared less well with 97% and 3% free and busy time. The G/G/1 model gives a better prediction than the M/M/1 model. However, the model is still a little optimistic regarding its forecasting of busy and free times.

5.2.10 Results - Correlation between inter-arrival and service times

Figure 5.3 shows the results of the autocorrelation test between the sequences of inter-arrival and service times. The number of lag points on the x-axis is equivalent to the number of observations (i.e. 331). Starting with the inter-arrival times, we can see that the lags at positions 0, 25 and 240 respectively cross the confidence interval. With only three values passing the confidence line (and the lag at position 0 is expected), there is insufficient evidence of seasonality in the values of inter-arrival times. For service times we observed a number of lags outside the confidence interval at positions 0 –10, 100 –120 and 160. While the correlation at lag 0 is expected, there is weak evidence of seasonality for lower and middle values of service times. Finally looking at the graphs of both inter-arrival and service times, we can see two lags at approximate positions 53 and 140 passing the confidence line. This suggests there is weak evidence of association between the two variables.

In addition to looking at the autocorrelation plots, we also performed tests on

Figure 5.3: Autocorrelation plots for inter-arrival and service times

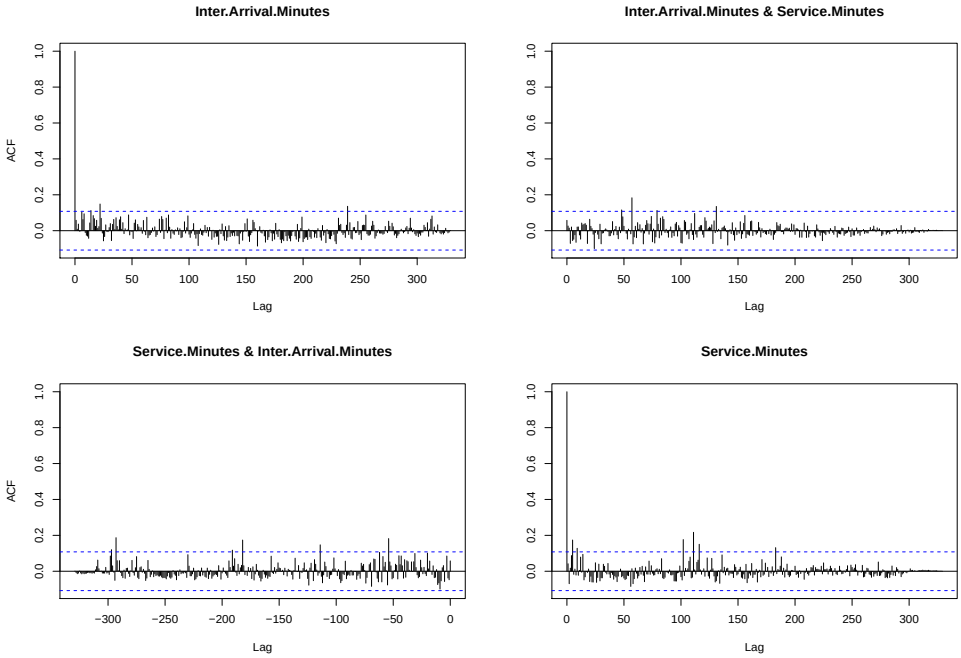
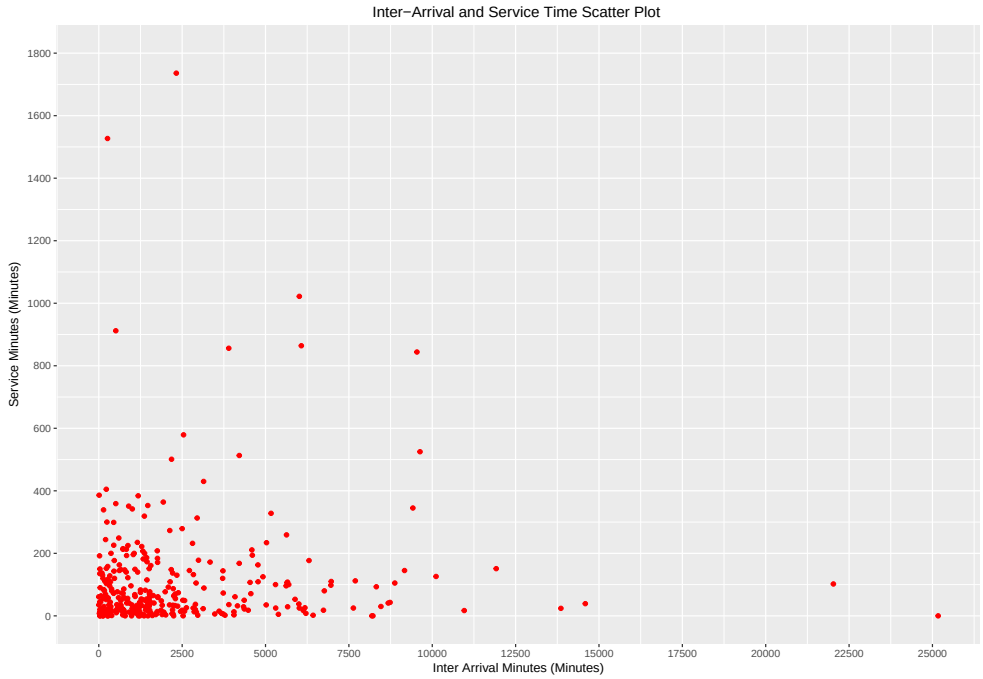


Figure 5.4: Inter-Arrival and Service Time Scatter Plot



corresponding pairs of inter-arrival times and services times to determine their relationship. Both Pearson and Spearman tests of correlation were executed. An r value were for each test was computed as follows: 0.06 (Pearson) and 0.06 (Spearman). These results indicate there is a very small positive correlation between inter-arrival and service times.

Finally, we ran a linear regression test using inter-arrival times as the dependent variable and service times as the independent variable. Our hypothesis states: There is no association between inter-arrival and service times. A p-value of 0.297 was computed. In other words, there is little evidence to suggest that inter-arrival and service times are correlated.

Figure 5.4 shows a scatter plot of the inter-arrival and service times plotted. A lack of linear or non-linear relationship is evident by the nebulous cluster of points on the graph.

5.2.11 Results - Assessment for no association and linkage between overlapping outage events

Analysis of the inter-arrival times between each of the 331 outages, was conducted to determine how many outage events overlapped. In other words, if an outage was currently being serviced by a DevOps resource, did a subsequent outage occur and if so where these overlapping events linked. Our analysis found 26 overlapping outage events. We inspected each outage report to determine if there was a link between these outages and outages already in the queue. As part of this study, we looked at the component affected and the root cause to determine whether a link between events was present.

We found evidence of a link (i.e. a common failure pattern) between 7 overlapping outages. It is worth noting that in 4 cases a temporal network outage was the root cause. In 5 cases the E-mail component was the component affected. While no formal regression analysis was conducted, we can conjecture that there is a correlation between Network failures and the E-mail component. Tables 5.5 and 5.6 contain additional details of this work.

Table 5.7 shows a two by two contingency table which contains counts of overlapping, non-overlapping, linked and non-linked outage events. Fisher's exact test was carried out on the table data. Our null hypothesis states there is

Table 5.5: Summary details of overlapping outages with analysis of component area, root cause and linkage assessment

Outage #	**Component**	**Root Cause**	**Outage Details**
1	E-mail	Network	Cascade network failures were observed in the email component. A second network failure was observed due to latency caused by the first network failure. Assessment: Outages linked.
2	E-mail	Network / Configuration	A network bottleneck was observed. A configuration change was made to alleviate the bottleneck, this change caused additional bottlenecks. Assessment: Outages linked.
3	E-mail	Concurrency	A failover operation failed to work correctly which caused an outage, while the system was in a failed state, crash log information was not output correctly. Assessment: Outages linked.
4	Social	High Availability	A number of nodes in the social component failed due to a server crash. While these nodes were down, extra load was added to the available nodes in the cluster which caused a subsequent outage. Assessment: Outages linked.
5	E-mail	Network	A temporal network outage occurred in the E-Mail system. Most of the nodes failed gracefully and returned to normal operations, however a number nodes did not fail gracefully which caused a secondary outage. Assessment: Outages linked.

Table 5.6: Summary details of overlapping outages with analysis of component area, root cause and linkage assessment (Continued)

Outage #	**Component**	**Root Cause**	**Outage Details**
6	E-mail	Configuration	A service on the E-mail system failed due to contention. A config change was made to remediate the initial contention. The config change caused additional contention further along the service stack. Assessment: Outages linked.
7	Collaboration	Network	A temporal network outage occurred in the collaboration component, which caused all nodes to fail gracefully, almost all nodes returned to normal when the network was restored. A number of nodes however were in a hung state (from the initial outage) which caused a secondary outage. Assessment: Outages linked.

Table 5.7: Test for no association between overlapping and linked outages using Fisher's exact test

Outage type	**Linked**	**Non-linked**
Non-Overlapping	0	305
Overlapping	7	19

Table 5.8: Summary of each research question, results and background literature

Research Question	Results	Literature
1. How are the inter-arrival times of Cloud outage events distributed?	Pareto distribution is the best fit. AD test statistic = 0.53 p-value = 0.72	Anderson-Darling goodness of fit test. [77]
2. How are the service times of Cloud outage events distributed?	log-normal distribution is the best fit. AD test statistic = 0.34 p-value = 0.90	Anderson-Darling goodness of fit test. [77]
3. How can an effective queuing model be built to simulate outage event traffic?	Our simulation achieved 94.3% busy & 5.7% free time compared to 92.1% free & 7.9% busy time for observed data.	Queuing theory. [9][10]
4. How are inter-arrival and service times correlated?	Weak evidence of association between inter-arrival and service times. p-value = 0.297	Ranked coefficients. [172][173] Linear regression. [87] Autocorrelation. [174]
5. Are overlapping outage events related or can we treat each event as independent?	Strong evidence of association between overlapping outage events. p-value <0.001	Fishers exact test for independence. [178][179]

no association between overlapping and linked outages. A p-value of <0.001 was calculated. Given the low p-value, we can reject the null hypothesis. In other words, based on our observations there is evidence to suggest that overlapping outages are linked to a common failure event.

5.2.12 Results - Summary

Before a detailed discussion of our results, we summarise the results along with each corresponding research question and relevance background literature. Table 5.8 provides this summary

5.2.13 Discussion - Inter-arrival time distribution

The results section has shown that the Pareto distribution is a good fit to model the inter-arrival times of Cloud outage events, which answers our first research question.

The decision to use Pareto as an inter-arrival time distribution is an interesting choice. The Pareto distribution is a power law distribution and has applications in many fields of science. However, the field where a Pareto distribution is typically used is in the area of finance. Specifically for modelling income and wealth [183].

The characteristics of our data that make Pareto so attractive is the number of values within a specific range. Inter-arrivals times range from 3 to 1057122 minutes. Using 2500 minutes as an arbitrary point of delineation, 71% of inter-arrival times were below 2500 minutes, while 29% were above 2500 minutes. While this split does not conform to the textbook "80-20" rule [184], it does illustrate that our data set contains a significantly higher proportion of shorter inter-arrival times than longer ones. Given this specific trait, it appears that the Pareto distribution is a resonable fit.

This study has answered our first research question: What distribution can be used to model inter-arrival times of Cloud outage events. DevOps teams can use the shape and scale parameters of their inter-arrival distribution to compute a mean and standard deviation, which can provide an expected time between outage events. Additionally, this result can be used to compute the proportion of inter-arrival times above or below a specific duration. These results can be used to aid resource planning. For example, if the expected inter-arrival time is known or if a high proportion of outages is known to occur within a specific duration, duty rosters can be generated to ensure adequate staffing is available when an outage occurs. Finally, this result can be used as a component in a wider queue model framework to infer team busy time.

5.2.14 Discussion - Service time distribution

We have learned from our results that the log-normal distribution is a reasonable fit to model the service times of Cloud outage events recorded in our dataset. We note that there is an extreme service time value that does not fit the distribution. Looking at this extreme value more closely, we indicate that

this service was over 24 hours in duration, and affected all components. We also note that this type of event occurred once over the 22 month period of data collected.

The log-normal distribution is significant in the description of natural events. Many natural occurring processes are modelled by the culmination of incremental changes. Such methods include general system usage, vehicle mileage per year, count of switch operations and wearout characteristics of machines and systems. This distribution is also versatile in that depending on the location and scale parameters many different distribution shapes can be accommodated.

In the context of our dataset, we can say that a log-normal distribution is useful to model outage service times observed, however, for extreme/rare outage durations a heavier tailed distribution may be more appropriate.

This result adds to the wealth of existing studies which support the notion that service times for repairable systems can be modelled using a log-normal distribution. We noted previously the work done by Kleyner and O'Connor [35]. However, a number of additional recent studies have observed similar results in their studies of repairable systems such as Apostolakis et al. [185], Ananda and Malwane [186] and Ananda et al. [187].

This study has answered our second research question: What distribution can be used to model service times of Cloud outage events. DevOps teams can employ the location and scale parameters of their service distribution to compute a mean and standard deviation, which can provide an expected duration service time. With the service times known, teams can create schedule plans to determine expected engagement times. Finally, this result can be used in conjunction with the result from the previous section to model the idle and busy times of a team as part of a queue modelling exercise.

5.2.15 Discussion - Outage event modelling framework

We asked the question: How can an effective queuing model be built to simulate outage event traffic? The result from our simulation shows that, by combining well-fitted distributions for both inter-arrival and service times a model can be built which provides a reasonable level of precision to simulating queue busy times compared to observed data.

Table 5.9: Sample output from our G/G/1 simulation

Duration (Minutes)	Queue Length	Date & Time
2	1	2017-01-01 00:02
142	0	2017-01-01 02:22
3744	1	2017-01-03 14:24
3761	0	2017-01-03 14:41
5577	1	2017-01-04 20:57
5644	0	2017-01-04 22:04
11043	1	2017-01-08 16:03
11048	0	2017-01-08 16:08
14989	1	2017-01-11 09:49
15186	0	2017-01-11 13:06
19566	1	2017-01-14 14:06
19605	0	2017-01-14 14:45
22249	1	2017-01-16 10:49
22278	2	2017-01-16 11:18
22286	1	2017-01-16 11:26

Table 5.4 provides a summary of the % busy and free time for our observed data and two sets of simulations: G/G/1 and M/M/1 queues. It is unsurprising that M/M/1 lacks precision. There are two factors to consider here. First that neither the inter-arrival nor service distributions could be adequately modelled using an exponential distribution. We recall from Table 5.2 and 5.3 the AD test statistic for the exponential distribution was infinite. For the simulation, we used the computed means from the Pareto and log-normal distributions. Second, the M/M/1 queue assumes that the arrival times between events are independent. We have shown that a small proportion (2%) of overlapping outage events are linked. These two factors make the M/M/1 queue unsuitable for queue simulation based on the observed data.

Conversely the G/G/1 provided a greater degree of precision than the M/M/1 queue. This is because the two distributions selected were a good fit against the observed data compared to the exponential distribution. There is still a lack of fidelity between our G/G/1 simulation and our observed data. We can surmise that while the goodness-of-fit for the service time distribution was very good (AD statistic = 0.34), the goodness-of-fit for the inter-arrival time distribution was good (AD statistic = 0.54). Moreover, there is the question of

independence between arrivals. We must conclude that with a small number of dependent outages coupled with the less than exact fit of the inter-arrival distribution may skew the precision of our simulation. Additionally, we also note that due to the underlying random process of simulation, our results may vary upon subsequent simulations. To understand a range of plausible values for queue busy and free times, multiple simulations could be run to obtain such a range of values. We discuss improvements to this model in our future work section in chapter 8.

Now let us look at the practical application of such a simulation model. The core of idea of this case study is to produce a model which is effective in simulating the arrival of Cloud events. We have previously mentioned the challenges that both micros teams and SMEs have when working in the area of Cloud computing. One of the key challenges is the deployment of resources, and how one can position these resources where they are most needed. Let us consider the following scenario as an example of our simulation framework.

Table 5.9 shows the output from a G/G/1 simulation using the parameters from our data set. There are two columns: Time (Measured in minutes) and Queue length. Let assume that we have uptime from 12:00 1st of January. Looking at the output below we can see that we will need one resource to service the first thirteen outage events. These thirteen outages will arrive and be serviced within sixteen days. Looking at the fourteenth outage event, we can see this event will arrive at approximately 11:18 on the 16th of January. DevOps Management has a good indication that two DevOps resources will be required at this time. One to service the thirteenth outage and a second resource to service the overlapping fourteenth outage. DevOps management can also infer that both resources will be required for only for a short duration. In this case eight minutes approximately. After that one resource will be needed to debug and remediate subsequent outage events.

Another application of the queue simulation model is to assess staffing requirements over a calendar year. By knowing the duration of a year in minutes (525600), we can easily check to see how many events will occur during a calendar year. In a simulation conducted for this example, we found the queue length was greater than one on 28 occasions. 27 times the queue length = 2 and once the queue length = 3. A final application is to look at the queue

busy time in a given calendar year. If we add the times the queue is busy (i.e. time difference between the queue length being 1 or more and 0) for outages that occur over the period of a year, we can see that the queue will be busy for approximately 144 days. These types of what if scenarios are very useful for resource planners.

5.2.16 Discussion - Correlation between inter-arrival and service times

Using a myriad of tests we have answered our fourth research question: How are inter-arrival and service times correlated? Our results show that there is little evidence to suggest a correlation between inter-arrival and service times.

Figure 5.3 showed graphically how both variables were correlated not only with themselves but each other. Given the low number of lags crossing the confidence interval coupled with the sparse positioning of these lags, there is little evidence to suggest any meaningful correlation. Likewise, we saw similar results from both the Pearson, Spearman and simple linear regression tests. Finally, Figure 5.4 provided additional graphical evidence of a lack of correlation between inter-arrival and service times.

DevOps teams can use this result in a number of ways. An ideal goal for a Cloud-based business is to have as near to 100% uptime as possible while ensuring that when an outage does occur, that the time to service such an outage is as short as possible. In other words, having very long inter-arrival times between outage events and very short service times is highly desirable. The goal for each requires a separate solution, in the case of long inter-arrival times ensuring that when a system does fail, it fails gracefully without any loss of service. In the case of service times, having an advanced suite of system monitoring solutions coupled with a simple system of rollback to prior code versions and configuration changes is key.

Given the lack of correlation between inter-arrival and service times, DevOps teams can be confident that process changes to reduce service times will not lead to a reduction in inter-arrival times. Moreover with increased reliability brings longer inter-arrival times, this, in essence, will not lead to longer service times.

5.2.17 Discussion - Assessment for no association and linkage between overlapping outage events

Our results section has highlighted there is an association between outage events that overlap and outages which are linked (i.e. cascade failures).

Tables 5.5 and 5.6 provide a good insight into the nature of linked outage events we saw that in five of the seven linked outages that E-mail was a common component. Likewise, we observed that network and configuration issues were the root causes in four of the seven outages, this may not be a coincidence. In one other case we saw that in a disaster recovery scenario, server node failover did not work as expected which caused a cascade failure due to high concurrency.

Table 5.7 highlights that overlapping outages are uncommon with approximately 8% of all outages recorded over an eighteen month period overlapped. Additionally linked outages are rarer still with only approximately 2% recorded over the same duration. However as demonstrated by the results of Fisher's test, there is overwhelming evidence to suggest that both events are associated. Therefore overlapping outage events are related. We can calculate when an overlapping event occurs there is an approximately 25% probability that these events are linked. Removal of these types of failures is key to the success of a business and will lead to increased customer satisfaction by increased up time.

DevOps teams can learn from these results; linked failures cause additional workloads for small teams. From a remediation perspective, DevOps teams can work with their software development counterparts to ensure their infrastructure and software are more resilient to temporal network outages. By conducting a series of negative tests, teams can determine how gracefully their systems fail under scenarios like temporal network outages. Additionally by setting invalid parameters within a large distributed system can have knock-on effects. It is worth pointing out that by introducing a system of managed configuration changes (similar to developer code reviews before check-in), can help alleviate the problems encountered with invalid configuration changes.

Finally, as we noted in our results section, overlapping outages are linked to frequent failure events, it may be expedient to periodically perform the

a Fisher test to determine if linked events are common without reading the outage reports in full.

5.3 Conclusion

One purpose of this research was to examine which probability distributions could be used to best model inter-arrival and service times of outages. By using the best fitting distributions as part of a special case of the G/G/1 queue modelling system, this study demonstrated how this model could be used to determine the busy time of a Cloud outage queue system. Additionally, this study examined the correlation between inter-arrival and service times. Furthermore, we observed whether overlapping outage events are linked.

It was found that inter-arrival and service times of Cloud outage events could be reasonably modelled with a Pareto and log-normal distribution respectively. Additionally, by using these distributions, a queue model framework could be built to infer the percentage busy time of this queue with a good degree of accuracy. Furthermore, we found no evidence of a correlation between inter-arrival and service times. Finally, our research showed that there is evidence to suggest that overlapping outage events are linked.

The findings of this study support previous work specifically in the field of repair times of maintainable Cloud-based software systems. This work provides more comprehensive analysis of the inter-arrival times of Cloud outage events and how using inter-arrival and service time distributions a useful special case of the G/G/1 can be developed to determine queue busy time.

However, the primary application of this research is to DevOps and project planners within an SME or micro team. We suggest a single queue system to model resource busy times as follows. With small teams, there is a higher likelihood of a developer or DevOps engineer owning the remediation of all component failures. With a single point of resolution, a single queue model appears reasonable. Conversely, a multi-server may be more appropriate for larger development teams or teams with a development resource per component.

Small teams can use the queue simulation technique to build an accurate resource planning model which can both identify skill and personnel gaps.

Identification and remediation of these gaps will significantly benefit teams in the challenging area of Cloud outage resolution.

In our next chapter, we consider the idea of how teams resolve problems within the context of real-time collaboration applications. For example, DevOps personnel are distributed, they may use such software to fix Cloud outage events. We look at how modelling group chat conversations (specifically in the domain of software development) may provide an analogue to service time remediation.

CHAPTER 6

Chat Discourse Modelling

As startups and micro teams adopt real-time collaborative instant messaging solutions, a wealth of data is generated from day to day usage. Making sense of this data can be a challenge to teams, given the lack of inbuilt analytical tooling. In this chapter, we model the distributions of duration, inter-arrival time, word count and user count of real-time electronic chat conversations in a framework, where these distributions can be used as an analogue to service time estimation of problem determination. Using both an enterprise and an open-source dataset, we answer the question of what distribution family and fitting techniques can be used to model real-time chat conversations for the purposes of understanding the expected time of a conversation. Our framework can help startups and micro teams alike to effectively model their real-time chat conversations to allow high-value decisions to be made based on their collaboration outputs.

6.1 Introduction

Real-time collaboration solutions are being marketed as a way for teams, regardless of size to increase their productivity [188] [189] [190]. One of the benefits of using such software is that conversations are segmented into either spaces, channels, or chat rooms which facilitate discussion in a linear fashion. As all conversations are recorded, rolling back to prior conversations can be done with the spin of a mouse wheel or the swipe of a screen. High-end collaboration suites also include an additional set of features such as a

file repository, knowledge management software and ability to screen share. A number of feature-rich solutions include: Watson Workspace [191], Slack [192], Microsoft Teams [193] and Azendoo [194], to name but a few.

One of the key selling points of real-time collaboration suites is the idea that real-time communication reduces the need for e-mail communication [195], thus solving the problem of 'e-mail paralysis' [196] which is the effect of having such a large volume of e-mail an individual is unable to communicate due to the sheer amount of messages. However, both micro-teams and startups face new challenges with the adoption of real-time collaboration software. As usage increases over time so does the volume of data. Furthermore, as current offerings offer little in the way of in-built analytical solutions, making sense of the growing volumes of collaboration data is important.

While micro-teams and startups have a number of key use cases, a growing trend is for development teams and DevOps alike to use real-time collaboration software to facilitate their ability to debug problems, otherwise known as problem determination [197] [198]. The time to debug and fix a problem is typically defined as the service time and the time between successive problems is known as the inter-arrival time. This work will allow us to explore whether there is a relationship between outage modelling and simulation discussed in chapter 4 and chapter 5 and real-time chat durations. Both of these concepts form part of the wider field of Queuing theory [9].

As DevOps and development teams use real-time chat software to diagnose and debug outage events, our core motivation for this work is to explore whether there is a relationship between the inter-arrival and duration times of group chat conversations and the inter-arrival and service times of Could outages. In real terms we recall from chapter 4 that a Pareto and log-normal distributions were found to be useful regarding modelling outage inter-arrival and service times based on the data set provided.

We investigate whether chat conversations can be modelled with similar distributions to those found in chapter 4. For this work, two data sets are provided. An enterprise dataset containing chat conversations that diagnosed the Cloud outages that were the basis of the dataset used in chapter 4. Our second dataset is an open source dataset. While this dataset has no relationship to the enterprise chat dataset, we analyse this dataset to understand if any

Table 6.1: Summary of dataset metrics and factors

Metric	**UbuntuDev-IRC dataset**	**Enterprise dataset**
Total # Messages	4223	3261
Duration (dd:hh:mm:ss)	3 days, 14 hours, 15 hours, 0 secs	200 days, 21 hours, 38 mins, 53 secs
Conversations annotated	231	312
Mean # messages per hour	49.1	0.68
# entangled onversations	57	27
Entangled conversation ratio	0.25	0.09

analysis result can be generalised from one dataset to another.

In this case study, we propose a framework that both startups and micro teams can use to effectively model their group chat instant messaging conversations using a number of available techniques. The core idea of this framework is for small teams to use the output of modelled conversations to gain insight into the expected time of a group chat and once a conversation has completed when the mean time until the next conversation begins. For startups and micro teams with limited team size, understanding the duration of a group chat conversation can aid problem resolution outcomes.

This study contains research conducted on two real-time chat discourse datasets. Our first dataset is an enterprise dataset from a real-time collaboration application; our second dataset is an open source data set from an Internet chat relay (IRC) channel. We investigate what techniques can be employed to effectively model the distributions of chat duration, interval, inter-arrival time, the number of words per chat conversation and the number of users per single chat conversation. Using the results of this study for our framework, a modelling suite can be developed to provide teams with a greater level of introspection of their chat data.

6.2 Case study 6 - Chat discourse modelling

Inter-arrival time modelling of social and collaboration message data has been shown to provide a useful way to make inferences about the underlying structure of message data [88, 89, 90, 91, 92, 93, 94]. We model both datasets

with the aim of allowing startups and micro-teams to infer the expected duration of chat conversations. This output is a useful analogue to service time determination.

The study presented in this paper examines 543 real-time chat conversations from two datasets. The details are summarised in Table 6.1.

The first dataset analysed was the open source Ubuntu dev IRC channel [199]. For our study, we reviewed approximately 4200 messages. For each message, we reviewed whether it was part of an existing conversation or part of a prior or subsequent conversation. For each unique conversation identified we assigned a numeric topic ID. As part of the review phase, we annotated 231 unique conversations. The total time period analysed was approximately 86 hours.

The second dataset analysed was from an enterprise instant message chat system which discussed cloud infrastructure problems. For our study, we reviewed approximately 3200 messages. For each message, we reviewed whether it was part of an existing conversation or part of a prior or subsequent conversation. For each unique conversation identified we assigned a numeric topic ID. As part of the review phase, we annotated 312 unique conversations. The total time period analysed was approximately 4820 hours.

Ideally, a chat conversation will start, progress then reach a logical conclusion. However, on occasion, an unrelated message will be injected into an existing chat conversation. We found a number of heterogeneous chat messages which appeared midway through a homogeneous chat conversation. We enumerated these 'entangled chat conversations' [200] in total 57 of the chat conversations from the Ubuntu IRC dataset and 27 conversations from the enterprise dataset were found to be entangled. We define 'entangled chat conversations' as conversations that contain more than one topic. While disentanglement of chat is an interesting conversation, we will focus for now on establishing more basic properties of the chat. We discuss chat disentanglement further in chapter 8, as part of our future work.

This study aims to answer the following questions. First, can the duration of our annotated chat conversations be modelled by a parametric method? If not can a non-parametric method be used? Second, can the durations between annotated chat conversations be modelled by a parametric method? If not

can a non-parametric method be used? Third, what is the most appropriate method to model the inter-arrival times of chat conversations? Fourth, what modelling techniques can be used to model the number of words and lines of text in a chat conversation? Fifth, to model the number of users present in a chat conversation, is a Poisson model appropriate?

6.2.1 Conversation duration modelling

We define conversation duration as the timestamp of the last message in a conversation subtracted from the timestamp of the first message in a conversation. A number of conversations were recorded as being zero minutes in length. This is due to a number of short (five messages or less) conversations completing in less than one minute.

Measuring the conversation duration is useful exercise given many teams use real-time chat collaboration software to discuss and debug problems. We can investigate whether conversation durations can be mapped directly to Cloud outage service times. In other words, could a log-normal distribution be used to model conversation durations or is a heavier or lighter-tailed distribution more appropriate? In the case of chat conversation duration times, our starting point is to conduct a parametric test to determine if a known distribution can be fitted to our data set. The benefit of attempting to fit a known distribution is that, if such a fit can be found, we can access the mathematical properties of such distributions (i.e. mean, variance, probability density function, cumulative density function etc.). When parametric methods fail to yield a useful result, additional methods can be employed (i.e. Distribution body and tail modelling [78, 79, 80], Hurdle methods [81] and non-parametric methods such as Kernel Density Estimation (KDE) [82, 83, 84, 85, 86])

For distribution fitting, we used the R package fitdistrplus [159] to fit various distributions to our dataset. To validate the efficacy of each distribution, the authors used the R package ADGofTest [160], which uses the Anderson-Darling goodness-of-fit test, to determine if the observed data follows a specific distribution [77]. This parametric approach will be carried out in subsequent sections of our study.

6.2.2 Conversation delta time modelling

We define conversation delta time as the time duration between chat conversations. For example, the timestamp of the starting message in a second conversation is subtracted from the timestamp of an ending messaging in a first conversation. It should be noted in the case of an entangled conversation; a conversation delta is recorded with a negative time value. While this may seem counterintuitive, we reason that while we had a mechanism to record the number of entangled conversations we also needed to measure the level of entanglement regarding time. By using the negative time, we can in effect determine at the glance which conversations are entangled.

Measuring and modelling conversation delta times can highlight the waiting time between prior and future discussions. These results can help answer questions about the expected time between conversations.

Due to the complex nature of the underlying data set (i.e. a mixture of logical conversation durations (positive durations) and entangled (negative durations), two techniques were considered. The first was to partition the dataset into positive and negative inter-arrival times. One subset contained the positive durations (from logical chat messages) and the second subset contained the negative durations (from the entangled chat messages). For distribution fitting, we used the absolute values for the entangled conversation subset. The second technique was to conduct KDE modelling on the entire conversation duration dataset.

Our approach to conducting a non-parametric test (KDE) to model the entire delta duration (both logical and entangled) was conducted using the R package Density [201].

6.2.3 Conversation inter-arrival time modelling

We define conversation inter-arrival time as the time duration between the start of a first chat conversation and the start of a second chat conversation. In other words, the inter-arrival time is essentially the sum of conversation duration plus conversation delta time.

Measuring and modelling the inter-arrival time conversation is beneficial. The inter-arrival time is an important component when combined with conversa-

tion duration since modelling the result can be used for to predict conversation busy and free times as part of a wider queue framework.

Also of interest is to understand whether the inter-arrival times of chat conversations can be modelled with a Pareto distribution, given this was the distribution found to be a reasonable fit to model Cloud outage events in chapter 4.

6.2.4 Conversation message & word modelling

A key component of any group chat conversation is the number of messages that are required to complete a conversation and the number of words used. Performing analysis on both variables can initially tell us if a distribution can be fitted to the data set. If a suitable distribution can be found, this result can help answer questions such as the expected number of lines and words in a chat conversation.

After that additional inference can be conducted such as topic and keyword analysis. However, both topic and keyword analysis is beyond the scope of this current work and will be discussed in more detail in chapter 7.

6.2.5 Conversation user count modelling

Conversation user count is defined as the number of unique users that contribute at least one message to a group chat conversation.

As in previous sections, if a suitable distribution can be found to fit user count data, this result can assist teams in determining the expected number of participants per chat conversation or the proportion of conversations that contain n number of users.

As we are dealing with count data with a small number of categories, our initial approach is to determine if a Poisson distribution is a suitable fit to our user count data. If there is sufficient evidence to suggest a lack of fit, a test for overdispersion and underdispersion will be conducted. If there is evidence to suggest some level of dispersion within our data, we shall employ a method of hurdle modelling. This model is then tested for goodness-of-fit.

To validate the goodness-of-fit of a Poisson distribution, the authors used the R package vcd [202], which uses the Chi-Squared goodness-of-fit test, to

determine the level of dispersion in our count data we used the R package AER [203].

6.2.6 Limitations of dataset

The dataset has a number of practical limitations, which are now discussed. The process of aggregating chat messages into a cohesive conversation is a subjective one. While every effort was made on the part of the authors to align messages to a thread, we accept that the process is subjective. Additionally, the post times for the Ubuntu chat were measured in hours and minutes only. As a result conversation duration, delta and inter-arrival times were recorded in minutes, whereas for the enterprise data set, these times were recorded in seconds.

The chat conversations that form part of this study are from a) an Ubuntu IRC developer channel and b) from an enterprise chat messaging system that discussed Cloud infrastructure problems. While we hope these examples will be representative of technical discussion channels, it seems unlikely they will be typical of all types of channels.

6.3 Results

We now explore the results of our analysis. Tables 6.3, 6.4 and 6.5 contain a summary of our results for easy reference.

6.3.1 Conversation duration modelling

Figure 6.1 shows a probability density function histogram for the enterprise dataset. A total of 55 conversations were found to be of 0 minutes in length (i.e. conversations that contained a single message utterance. In other words a first message was sent from an individual and no further on-topic replies were made after that). These values were removed from the dataset, and a Weibull distribution was found to be the best fit for the remaining 257 samples. An Anderson–Darling test statistic and p-value were computed as 1.1 and 0.31 respectively. The p-value is above the 0.05 significance level.

Figure 6.2 shows four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots. Looking at the Q-Q plot, we can see that

Figure 6.1: Enterprise conversation duration with fitted Weibull curve (Breaks every 200 minutes)

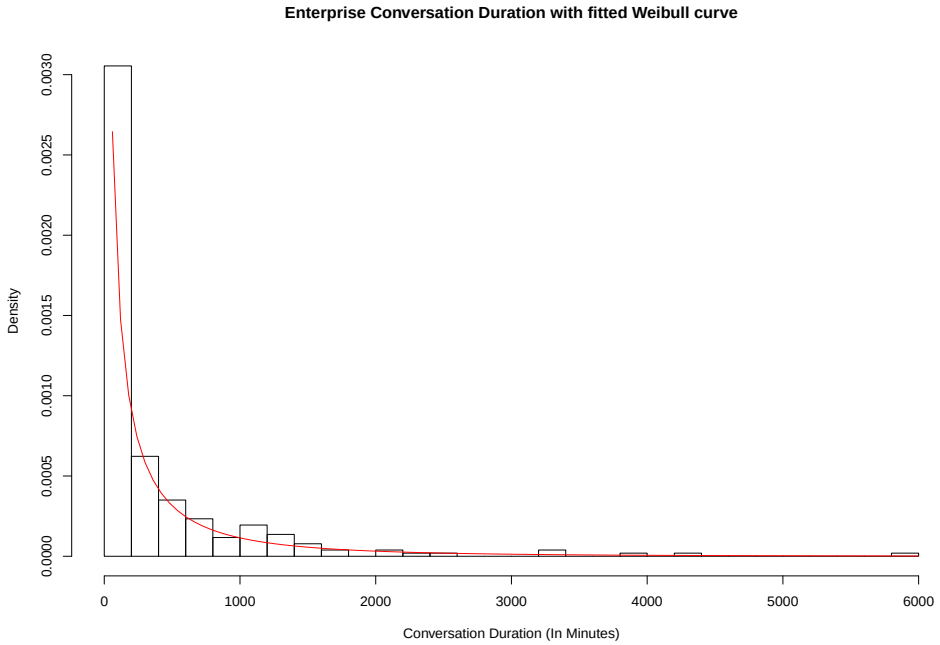


Figure 6.2: Four goodness-of-fit plots for Weibull distribution fitted to Enterprise conversation duration

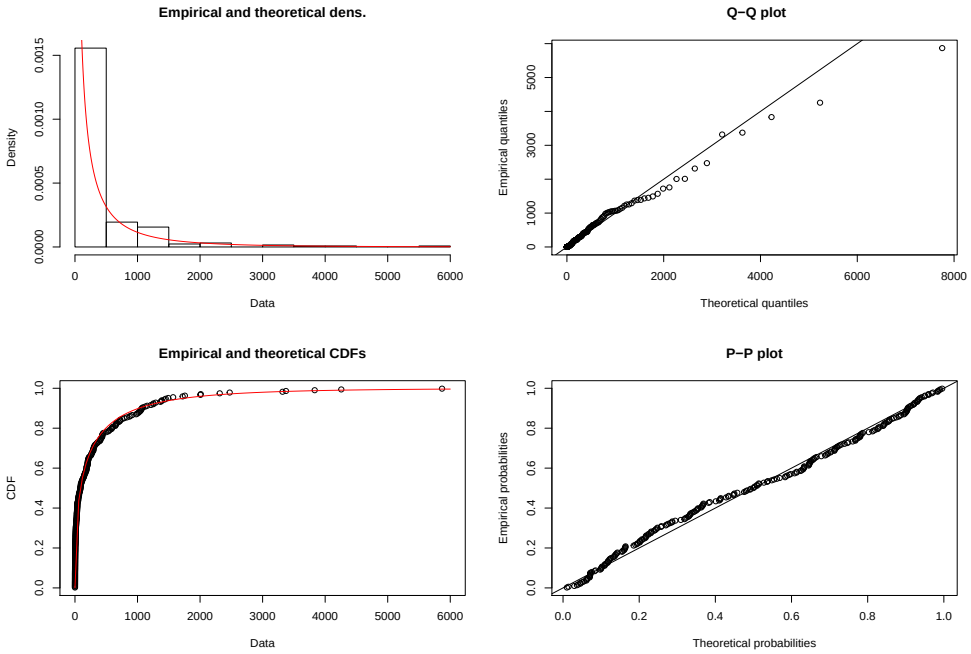


Figure 6.3: Ubuntu conversation duration with fitted Burr log-logistic curve (Breaks every 10 minutes)

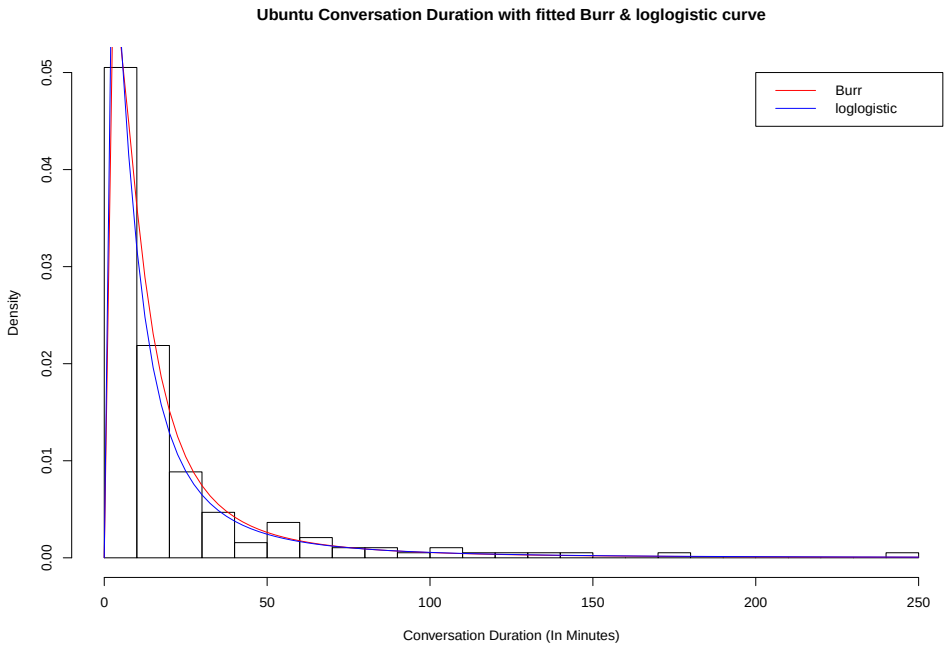
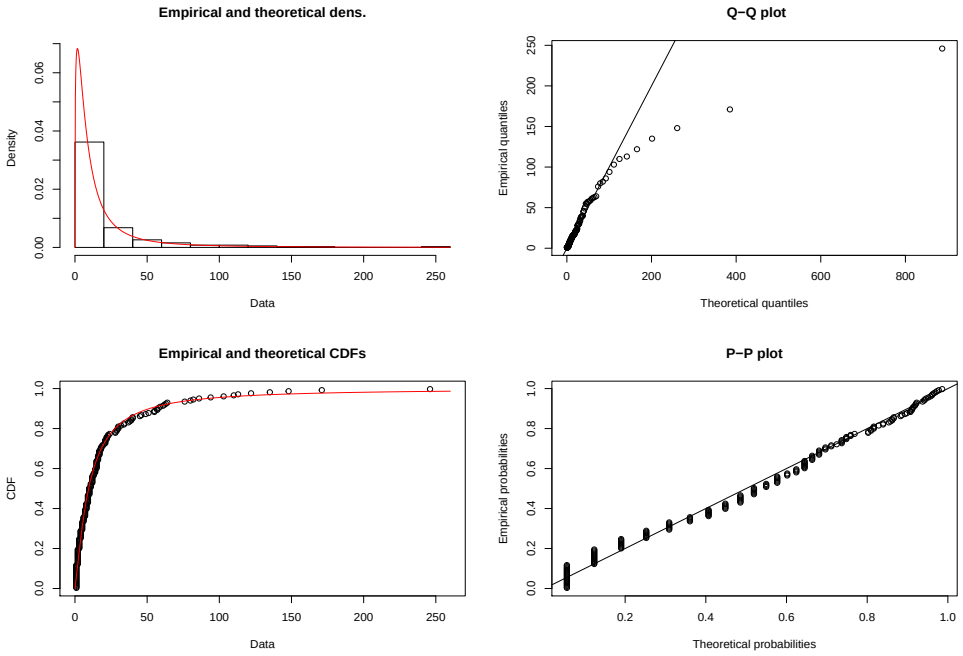


Figure 6.4: Four goodness-of-fit plots for log-logistic distribution fitted to Ubuntu conversation duration



for smaller quantiles the plotted quantiles fit the Weibull line reasonably well. However, for larger quantile points the values are more distant from the line.

Figure 6.3 shows a probability density function histogram for the Ubuntu dataset. For this dataset 39 conversations were found to be 0 minutes in length. Once again these values were removed from the dataset. Both a Burr and log-logistic distribution were found to be the best fit for the remaining 192 samples. An Anderson–Darling test statistic and p-value were computed for both distributions. The test statistic and p-value were the same for both distributions as 1.3 and 0.61 respectively. The p-value is above the 0.05 significance level.

Figure 6.4 shows four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots for the log-logistic distribution fitted to the Ubuntu dataset. Looking at the Q-Q plot, we can see that for smaller quantiles the plotted quantiles fit the log-logistic line reasonably well. However, for larger quantile points, seven values diverge from the line. The P-P plot also shows the plotted probabilities; the plotted points appear somewhat discrete for small values.

6.3.2 Conversation delta time modelling

The conversation delta time modelling results are split into two parts. The first is a parametric approach using MLE. In this approach the conversations were divided into two subsets: logical conversations (i.e. time duration between the end of an nth and the start of an nth+1 conversation, which is positive), and entangled conversation (i.e. time between the end of an nth and the start of an nth+1 conversation, which is negative). A second approach is a non-parametric approach using KDE.

Figure 6.5 and figure 6.7 show probability density function histograms of both the entangled and logical conversation delta times for the enterprise dataset. A Weibull distribution was found to be the best fit for both sub-datasets. An Anderson–Darling test statistic and p-value was computed for both distributions as 0.49 & 0.76 (logical dataset) and 0.3 & 0.94 (entangled dataset).

Figure 6.6 and figure 6.8 shows four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots. Looking at the Q-Q plot for the logical data set, we can see that for smaller quantiles the plotted

Figure 6.5: Enterprise logical conversation delta, with fitted Weibull curve (Breaks every 500 minutes)

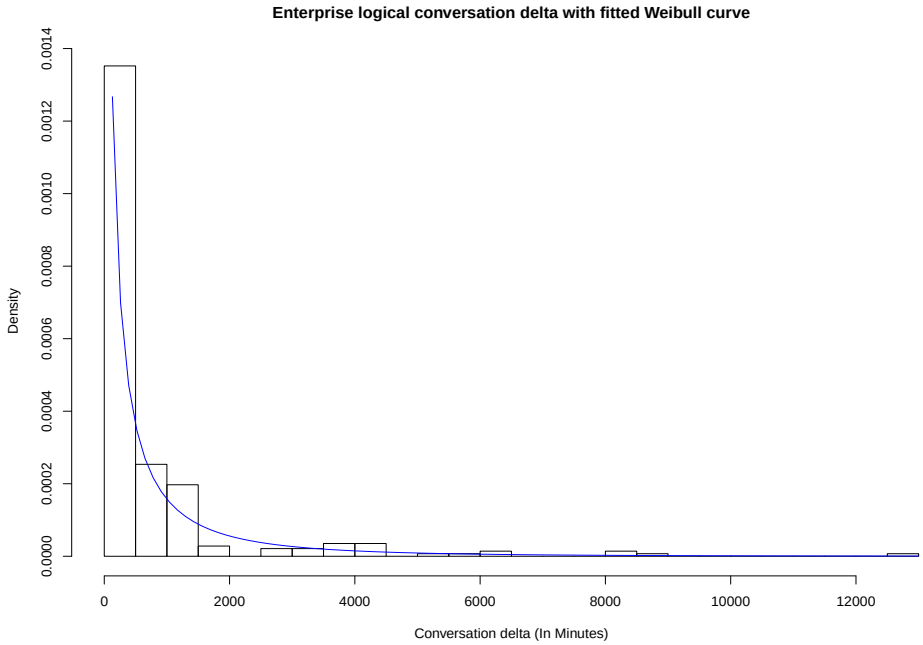


Figure 6.6: Four goodness-of-fit plots for Weibull distribution fitted to Enterprise logical conversation delta

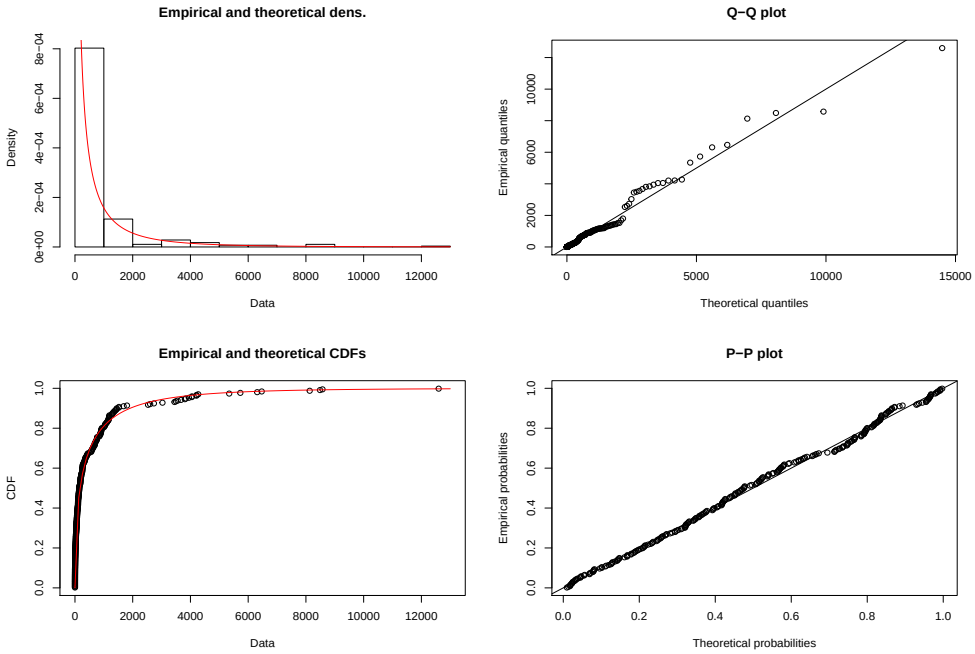


Figure 6.7: Enterprise entangled conversation delta with fitted Weibull curve (Breaks every 100 minutes)

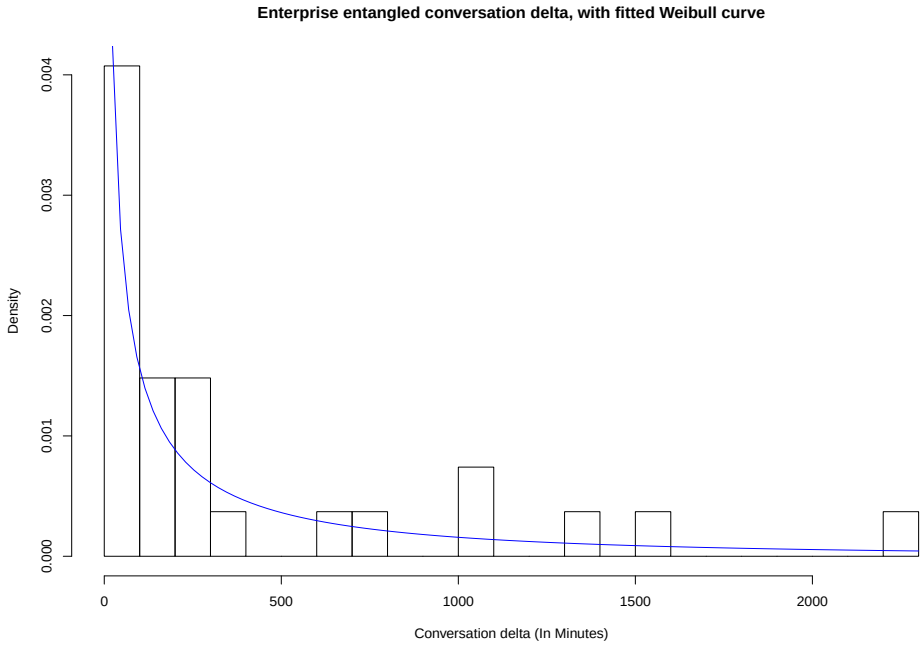


Figure 6.8: Four goodness-of-fit plots for Weibull distribution fitted to Enterprise entangled conversation delta

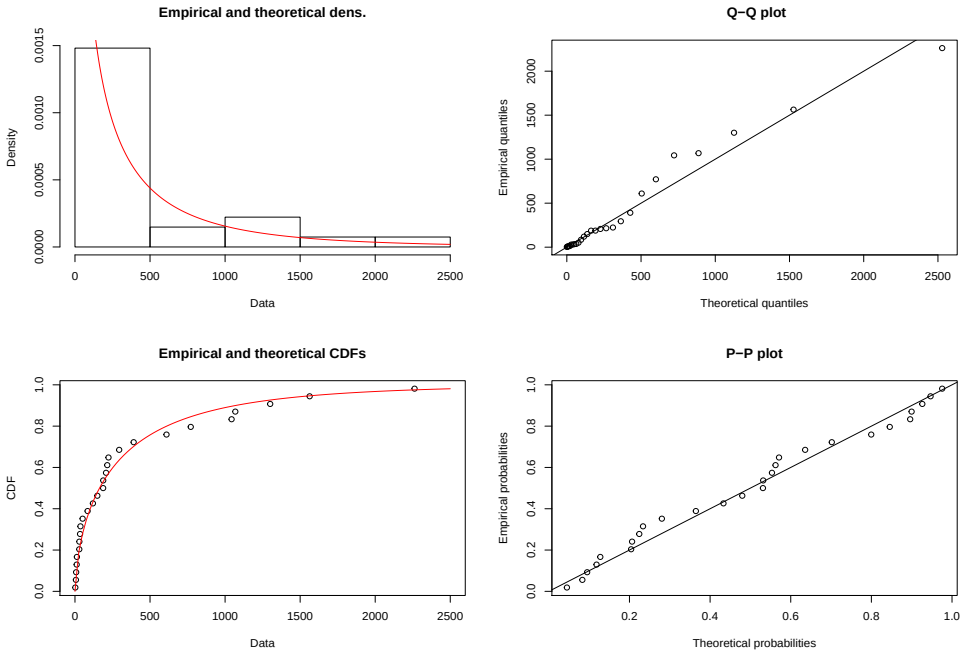


Figure 6.9: Enterprise conversation delta with fitted uniform (rectangular) kernel SJ-DPI bandwidth selection curve (Breaks every 200 minutes)

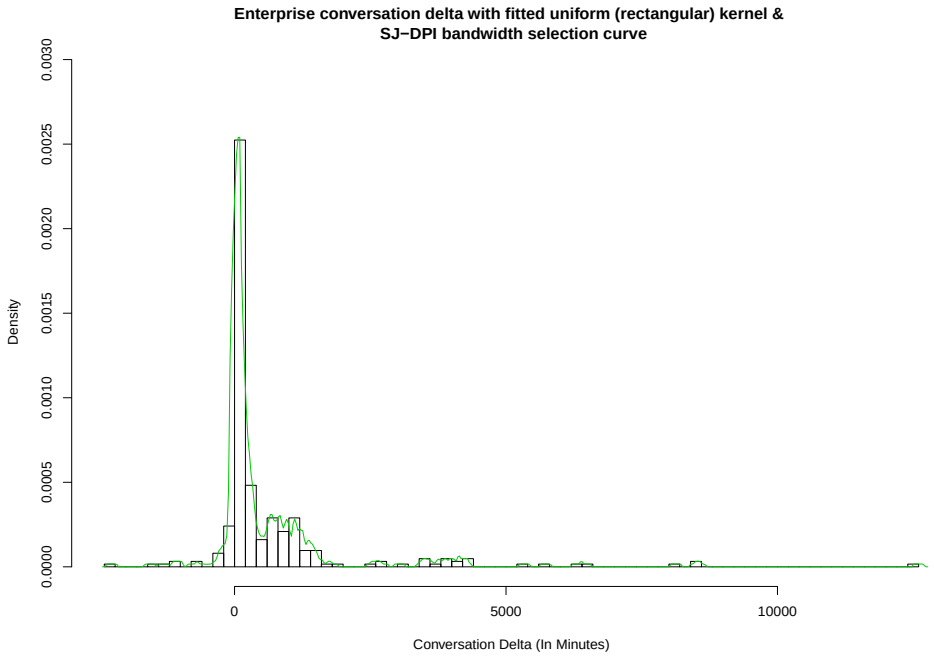


Figure 6.10: Ubuntu logical conversation delta, with fitted log-logistic curve (Breaks every 10 minutes)

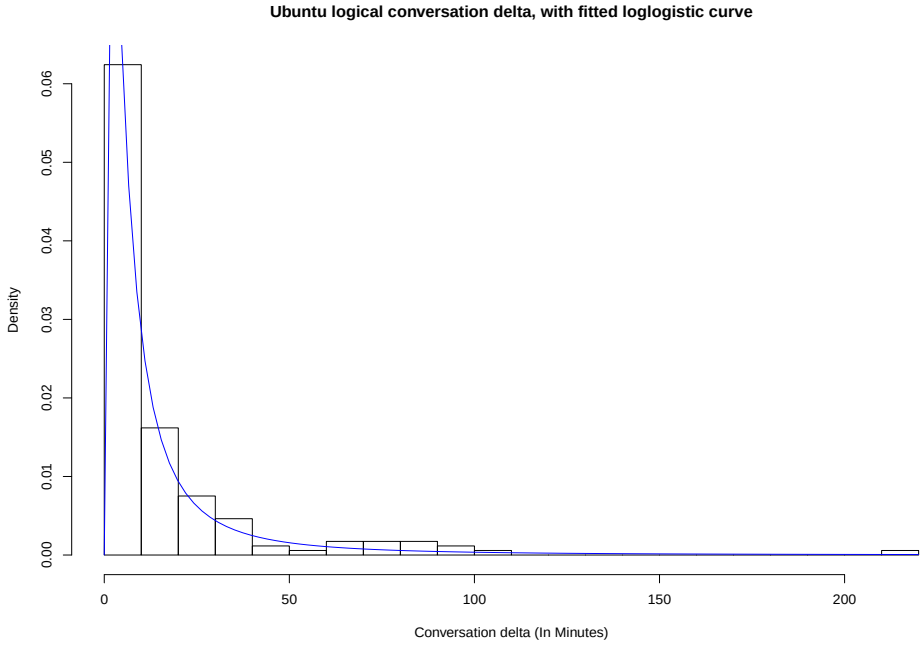


Figure 6.11: Four goodness-of-fit plots for log-logistic distribution fitted to Ubuntu entangled conversation delta

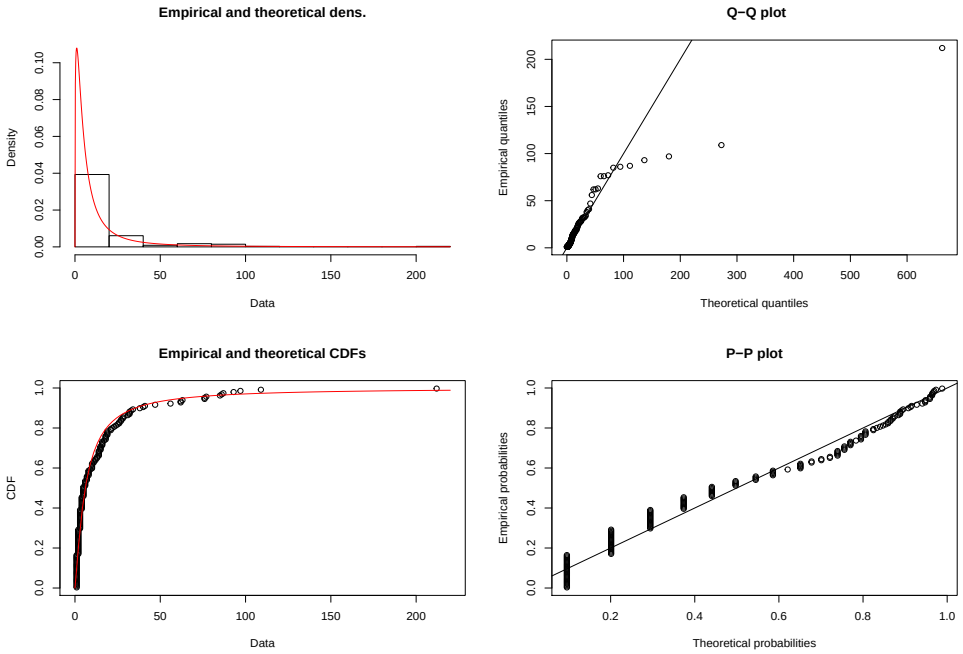


Figure 6.12: Ubuntu entangled conversation delta with fitted log-logistic curve (Breaks every 10 minutes)

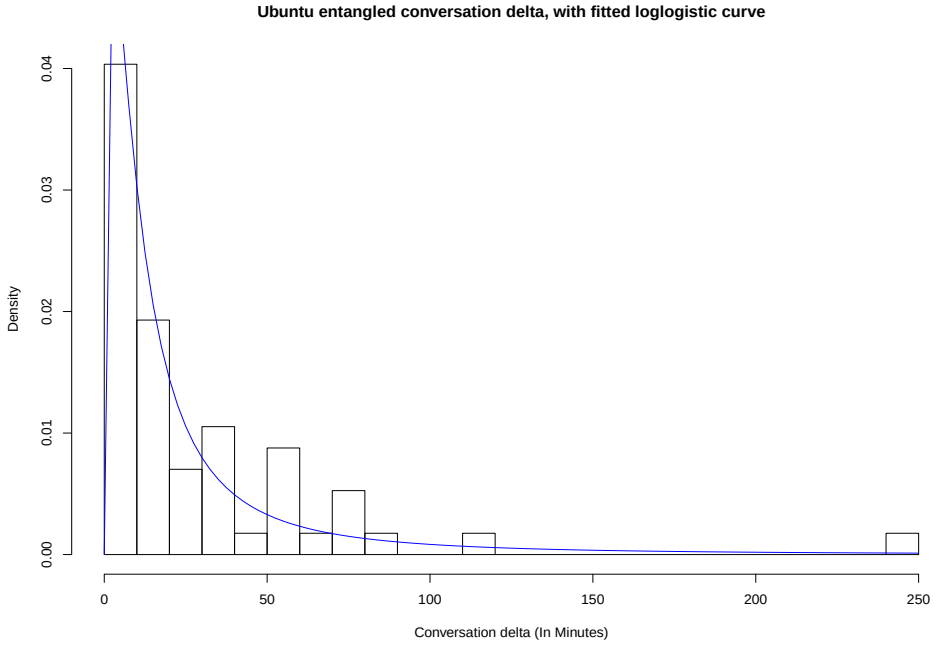


Figure 6.13: Four goodness-of-fit plots for log-logistic distribution fitted to Ubuntu entangled conversation delta

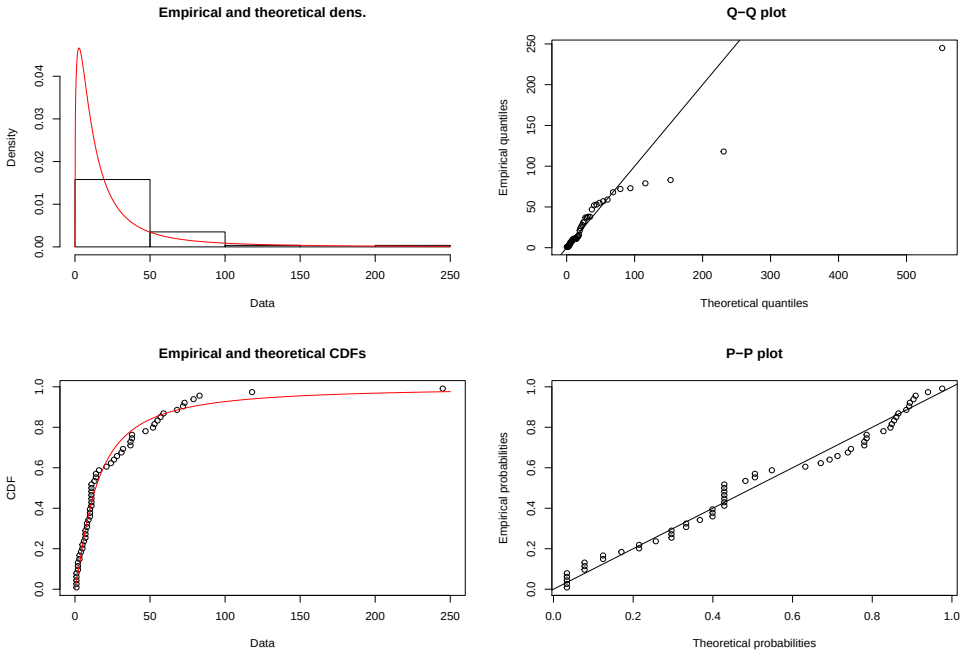
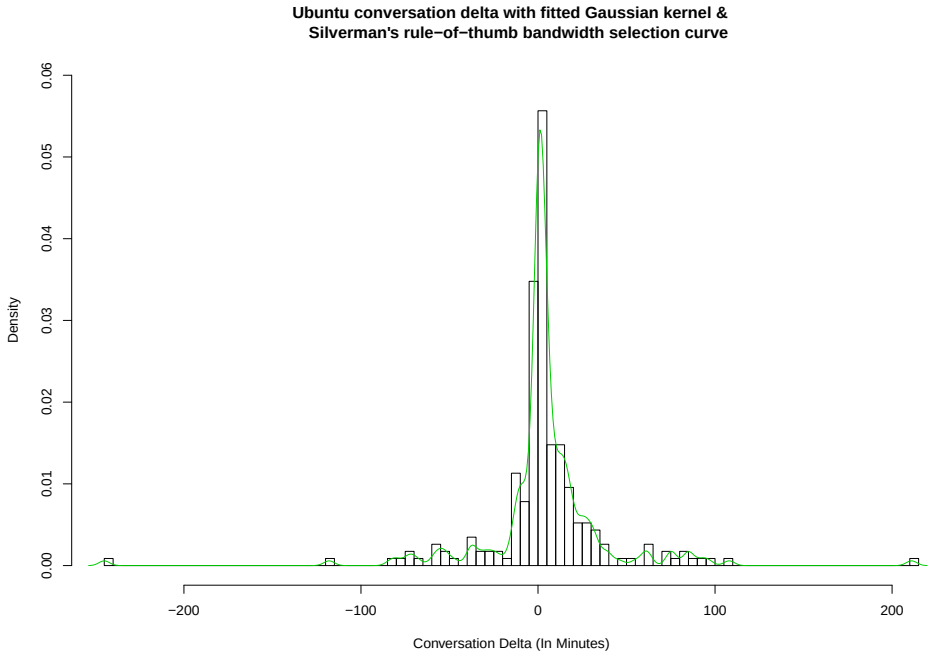


Figure 6.14: Ubuntu conversation delta with fitted Gaussian kernel Silverman's rule-of-thumb bandwidth selection curve (Breaks every 5 minutes)



quantiles fit the Weibull line reasonably well. However, for medium quantiles, there is a curve in the plotted quantiles, for larger quantile points the values are more distant from the line. Looking at the entangled dataset, for smaller quantiles, the fit is reasonable, for medium plotted quantiles, the pattern is more curved. The second largest quantile fits the line reasonably while the largest quantile is some distance from the Weibull line.

Figure 6.9 shows the output of a histogram of the combined entangled and logical conversation delta times for the enterprise data set. The Sheather–Jones direct plugin bandwidth selector combined with a uniform (rectangular) shaped kernel was found to be the optimal fit. The bandwidth was computed as h = 56.73.

Figure 6.10 and figure 6.12 show probability density function histograms of both the entangled and logical conversation delta times for the Ubuntu dataset. A transformation of one minute was added to the dataset to aid in the fitting of log type distributions. Zero values generated an undefined result in the fitting tool. Our remedial action added a small constant of one minute to each value in the dataset that allowed for a log type distribution to be fitted. A log-logistic distribution was found to be the best fit for both subsets. An Anderson–Darling test statistic and p-value were computed for both distributions as 2.46 & 0.05 (logical dataset) and 0.60 & 0.64 (entangled dataset).

Figure 6.11 and figure 6.13 shows four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots for the logical and entangled conversation delta for the Ubuntu dataset. Looking at the Q-Q plot for the logical data set, we can see that for smaller quantiles the plotted quantiles fit the log-logistic line reasonably well. However, for medium quantiles, there is a curve in the plotted quantiles, the largest six quantiles, appear away from the line. The greater the plotted quantile, the greater the distance from the line. The P-P plot illustrates that the data appears quite discrete for small values of plotted probabilities. The entangled (also a log-logistic distribution) shows a similar story to the logical dataset. For the Q-Q plot the majority of plotted quantiles fit the distribution line, while six quantiles away from the log-logistic line. Additionally, the P-P plot shows that the plotted probability points lie very close to the line.

Figure 6.14 shows the output of a histogram of the combined entangled and

Table 6.2: Summary of conversation delta time modelling results using a parametric approach or the Ubuntu dataset

Distribution	**AD Test Statistic**	**p-value**
Cauchy	3.486	0.0156
Log-normal	7.669	0.0002
Logistic	10.920	3.466e-06
Log-logistic	13.920	2.609e-06
Gamma	20.035	2.609e-06
Gumbel	38.655	2.609e-06
Exponential	85.705	2.609e-06
Burr	101.530	2.609e-06
Weibull	725.570	2.609e-06

logical conversation delta times for the Ubuntu data set. Silverman's rule-of-thumb bandwidth selector combined with a Gaussian shaped kernel was found to be the best fit. The bandwidth was computed as h = 2.94.

Table 6.2 provides a summary of the delta time modelling using a parametric approach for the Ubuntu dataset. As can be seen, none of the nine distributions tested provided an adequate fit to the overall delta time distribution. The Cauchy distribution came closest to fitting the delta time distribution. This result is due to it's probability density function (PDF) having a symmetrical shape and heavy tails. All other distribution types were easily rejected due to their high test statistic values.

6.3.3 Conversation inter-arrival time modelling

Figure 6.15 shows a probability density function histogram for the enterprise dataset. A Weibull distribution was found to be the best fit. An Anderson–Darling test statistic and p-value were computed as 0.78 & 0.5 respectively.

Figure 6.16 shows four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots. Looking at the Q-Q plot, we can see that for smaller quantiles the plotted quantiles fit the Weibull line reasonably well. However, for medium quantile points there is a slight kink in the line while larger quantile values are parallel and slightly offset from the Weibull line.

Figure 6.15: Enterprise conversation inter-arrival times with fitted Weibull curve (Breaks every 500 minutes)

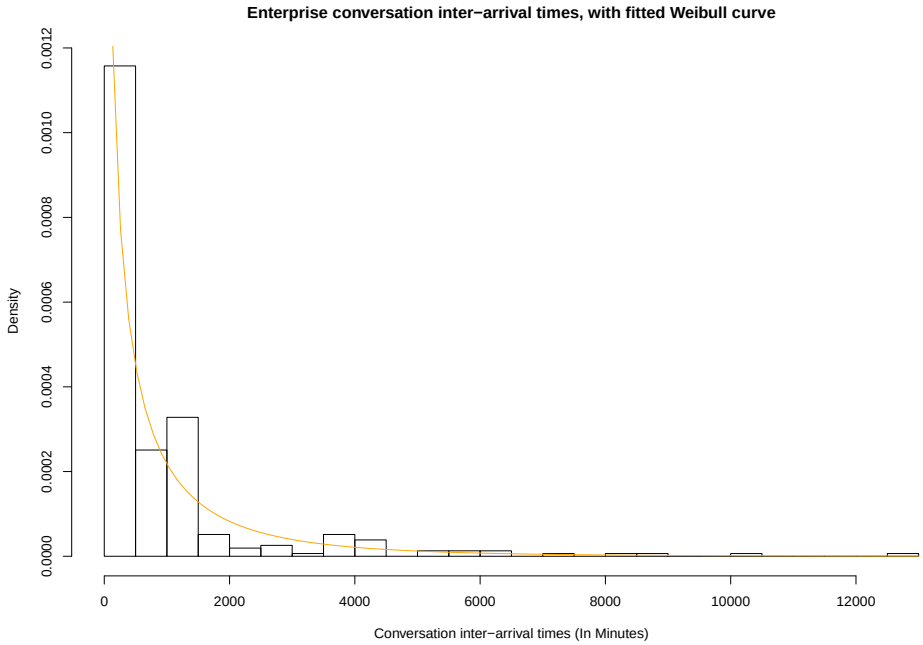


Figure 6.16: Four goodness-of-fit plots for Weibull distribution fitted to Enterprise conversation inter-arrival times

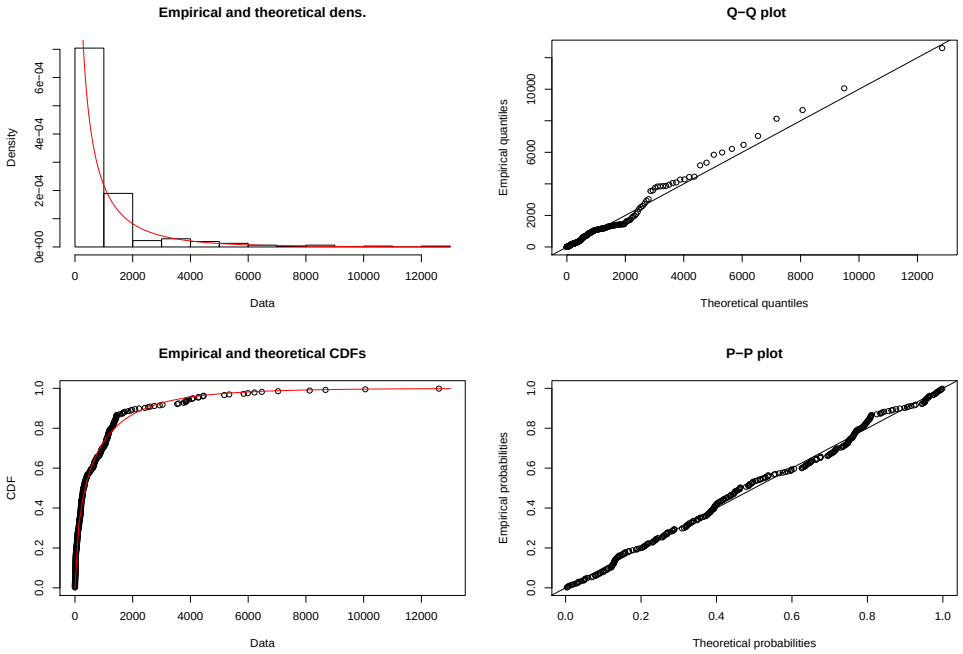


Figure 6.17: Ubuntu conversation inter-arrival times with fitted log-logistic curve (Breaks every 10 minutes)

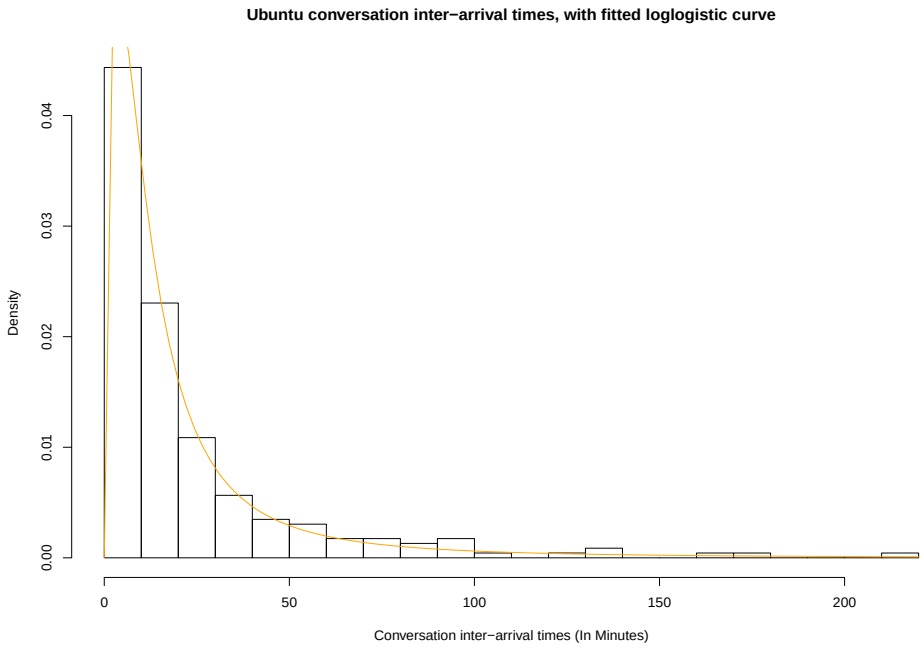


Figure 6.18: Four goodness-of-fit plots for log-logistic distribution fitted to Ubuntu conversation inter-arrival times

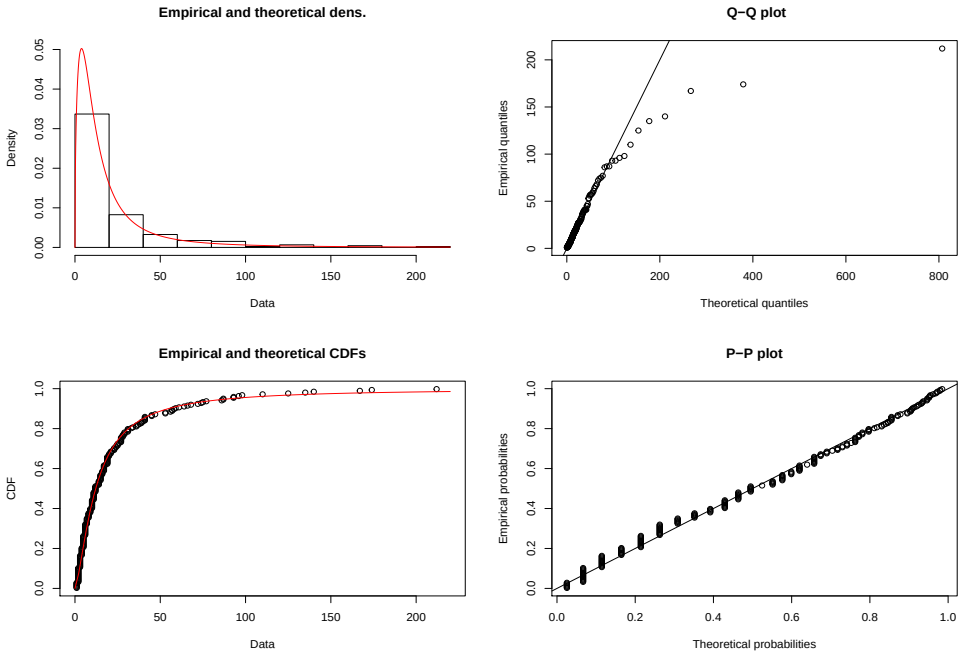


Figure 6.19: Enterprise messages per conversation with fitted Burr curve (Breaks every 5 messages)

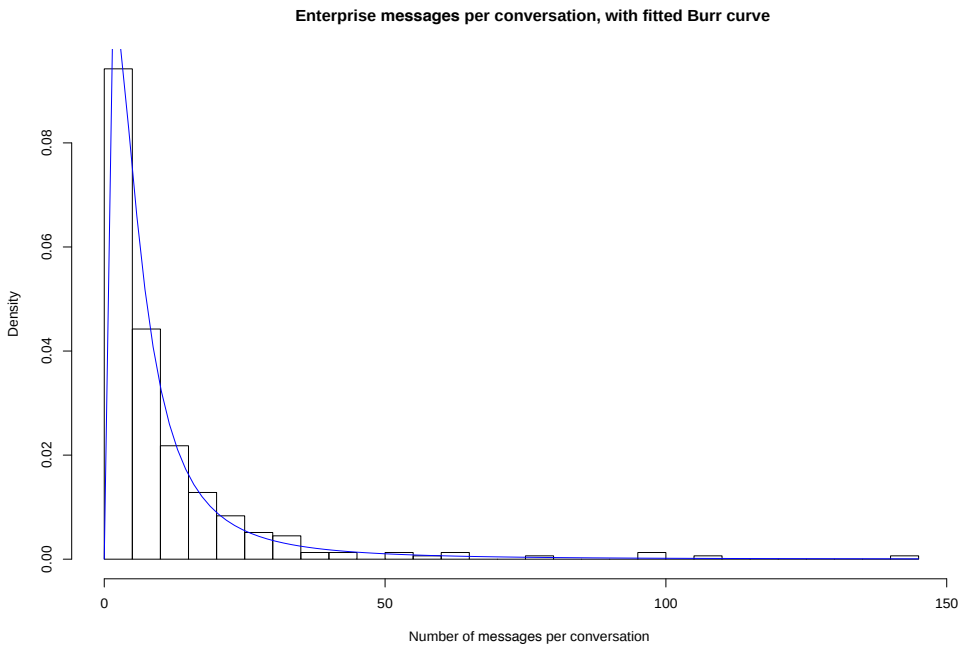


Figure 6.17 illustrates a probability density function histogram for the enterprise dataset. A small constant (1 minute) was applied to each value in the dataset. A log-logistic distribution was found to be the best fit. An Anderson–Darling test statistic and p-value were computed as 0.72 & 0.54 respectively.

Figure 6.18 illustrates four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots for the conversation inter-arrival times for the Ubuntu dataset. Looking at the Q-Q plot, we can see that for smaller quantiles the plotted quantiles fit the log-logistic line reasonably well. However, for the ten most extreme quantile points they move further away from the log-logistic line as the quantile points increase. The P-P plot shows the plotted empirical vs theoretical positions broadly fit the log-logistic line.

6.3.4 Conversation messages & word modelling

Figure 6.19 and figure 6.21 show probability density function histograms of both the messages and words per conversation for the enterprise dataset. A Burr distribution was found to be the best fit for messages per conversation. A log-logistic distribution was determined to be the best fit for words per

Figure 6.20: Four goodness-of-fit plots for Burr distribution fitted to Enterprise messages per conversation

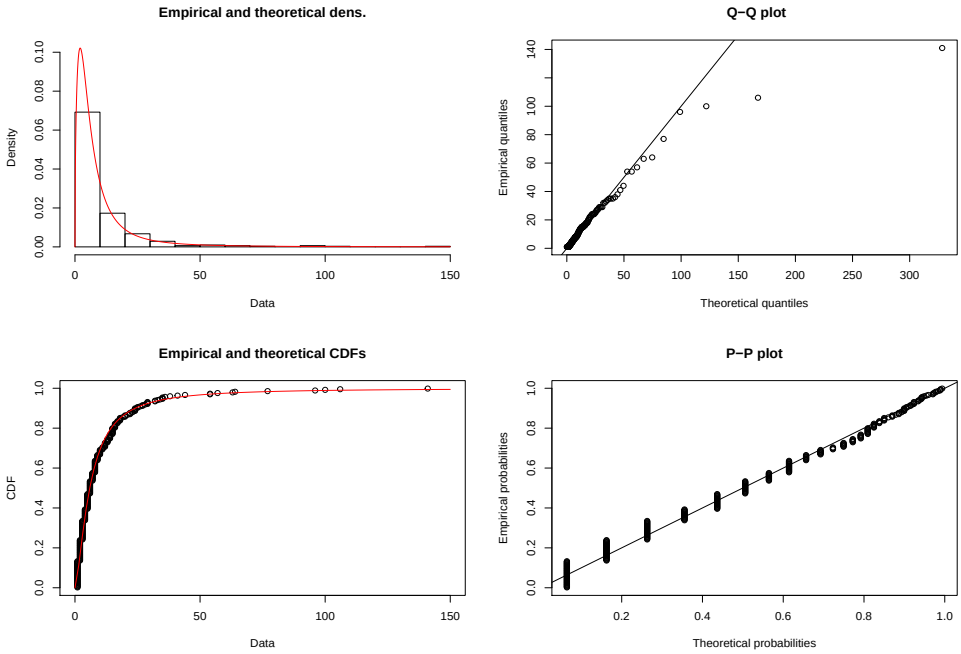


Figure 6.21: Enterprise words per conversation with fitted log-logistic curve (Breaks every 100 words)

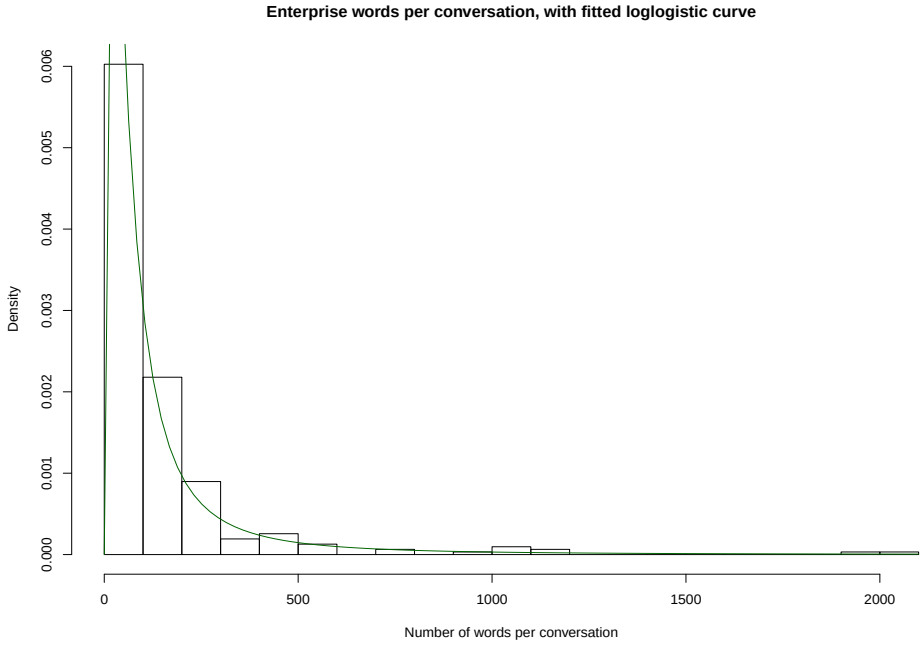


Figure 6.22: Four goodness-of-fit plots for log-logistic distribution fitted to Enterprise words per conversation

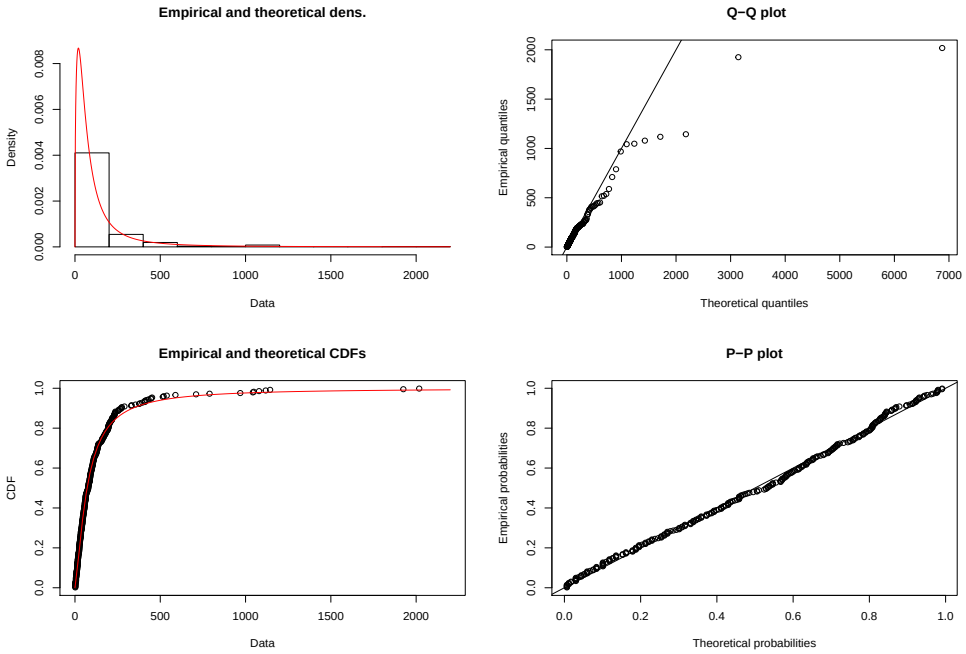


Figure 6.23: Ubuntu messages per conversation with fitted Burr curve (Breaks every 10 messages)

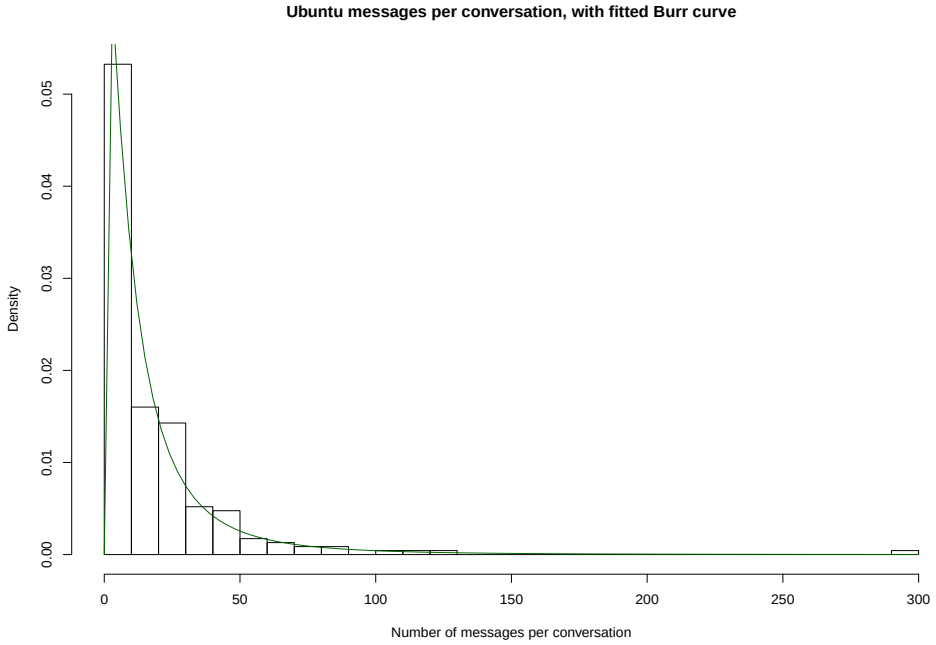


Figure 6.24: Four goodness-of-fit plots for Burr distribution fitted to Ubuntu messages per conversation

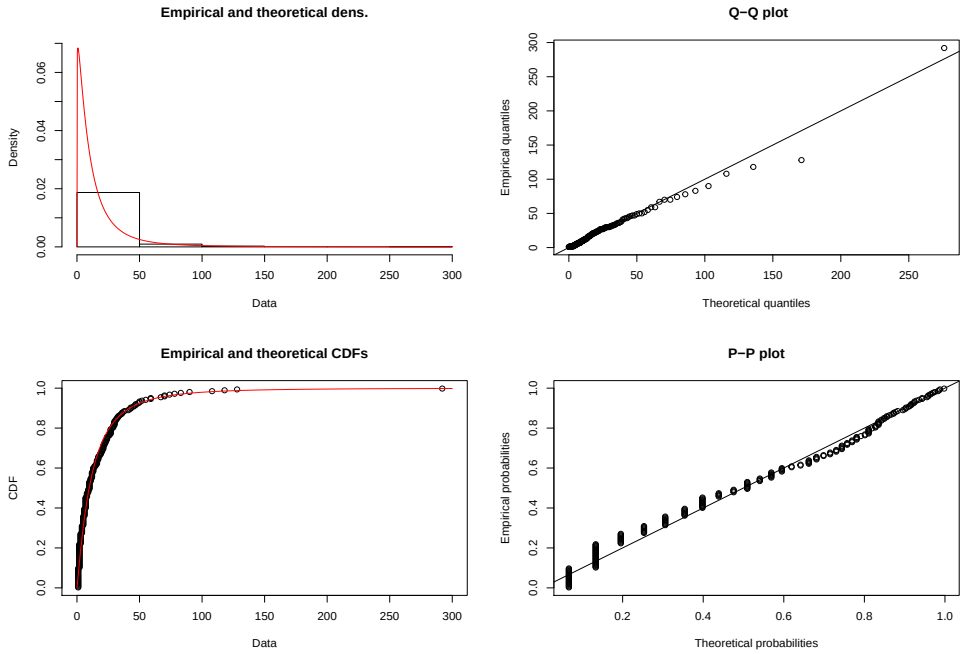


Figure 6.25: Ubuntu words per conversation with fitted Burr curve (Breaks every 50 words)

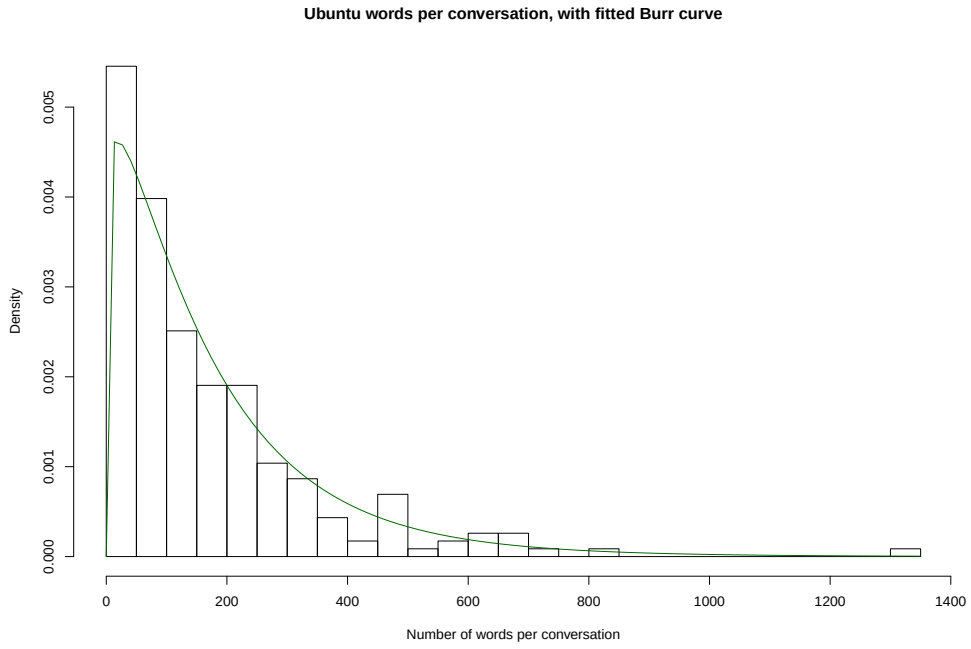
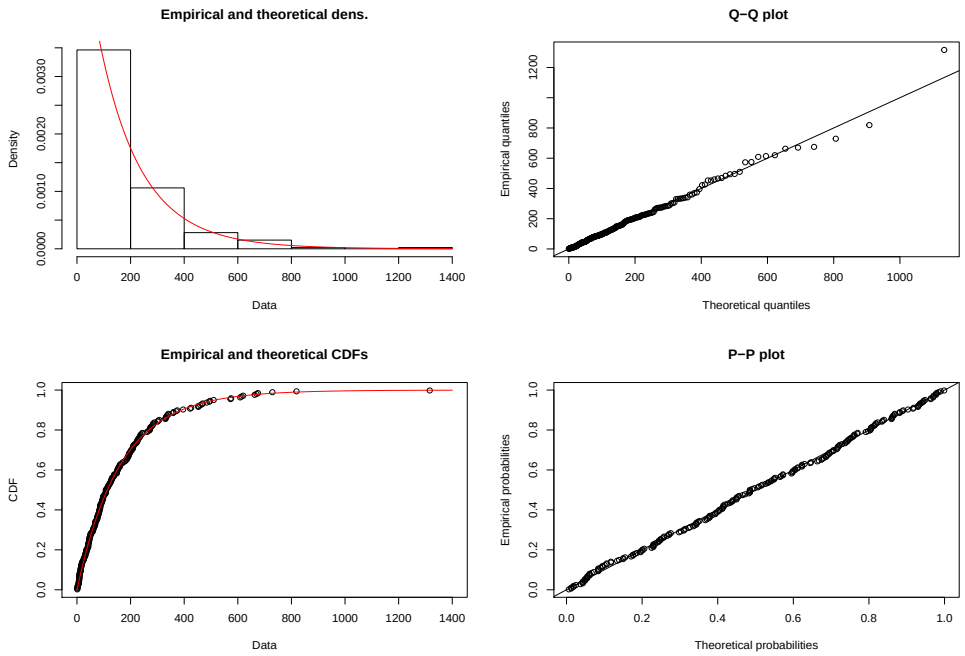


Figure 6.26: Four goodness-of-fit plots for Burr distribution fitted to Ubuntu words per conversation



conversation. An Anderson–Darling test statistic and p-value were computed for both distributions as 2.13 & 0.08 (messages per conversation dataset) and 0.65 & 0.6 (words per conversation dataset).

Figure 6.20 and figure 6.22 shows four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots for both the message and words per conversation for the enterprise dataset.

Looking at messages plots first, the Q-Q plot, we can see that for smaller quantiles the plotted quantiles fit the Weibull line reasonably well. However, for larger quantile points the values are more distant from the line. Looking at the messages graphs, we can see the Q-Q plot shows that the plotted empirical and theoretical quantiles fit the Burr line reasonably well. For extreme quantiles (i.e. the three largest), these quantiles are some distance from the Burr line. Additionally, the P-P plot shows the discrete nature of the data with plotted probabilities far apart for lower computed values, while more closely clustered for higher probability values.

Next looking at the the word goodness-of-fit plots for the enterprise dataset,

the Q-Q shows a reasonable fit to the log-logistic line for small and medium quantiles, however for more extreme quantiles (i.e. the six largest), these plotted points lies away from the line. The P-P plot shows no signs of a poor fit. All plotted points appear along the line.

Figure 6.23 and figure 6.25 show probability density function histograms of both the messages and words per conversation for the Ubuntu dataset. For both datasets, a Burr distribution was found to be the best fit. An Anderson–Darling test statistic and p-value were computed for both distributions as 1.76 & 0.13 (messages per conversation dataset) and 0.31 & 0.93 (words per conversation dataset).

Figure 6.24 and figure 6.26 shows four goodness of fit plots (histogram with a fitted distribution curve, Q-Q, CDF and P-P plots for both the message and words per conversation for the Ubuntu dataset.

Looking at messages plots first, the Q-Q plot, we can see that for smaller quantiles the plotted quantiles fit the Burr line reasonably well. There are seven large quantiles plotted values that appear off the Burr line. The P-P plot illustrates that the plotted probability points generally intersect the Burr line. However, there is a slight curvature between 0.6 and 0.8.

Next looking at the word goodness-of-fit plots for the Ubuntu dataset, the Q-Q shows a reasonable fit to the Burr line for small and medium quantiles, however for more extreme quantiles (i.e. the four largest), these plotted points lies away from the line. The P-P plot shows no signs of a poor fit. All plotted probability points appear along the line.

6.3.5 Conversation user count modelling

Figure 6.27 illustrates the PDF and cumulative density function (CDF) plots of user counts per conversation for the enterprise dataset. However, using the raw counts, a Poisson distribution was found to be a poor fit due to under-dispersion within the dataset. The level of dispersion was calculated as 0.75, which indicates some degree of under-dispersion (a value of greater than 1 would indicate over-dispersion within the data). A hurdle method was implemented whereby the counts of $n-1^{\text{th}}$ users were modelled. A chi-squared test statistic, degrees of freedom and p-value were calculated with the hurdle

Figure 6.27: Enterprise $n-1$ users per conversation with fitted Poisson PDF and CDF

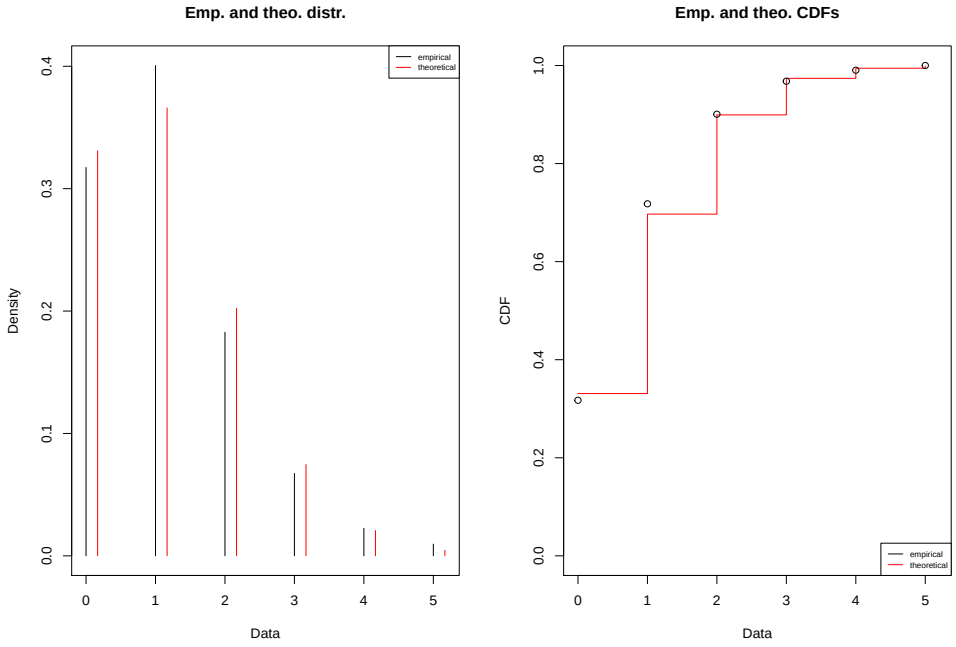
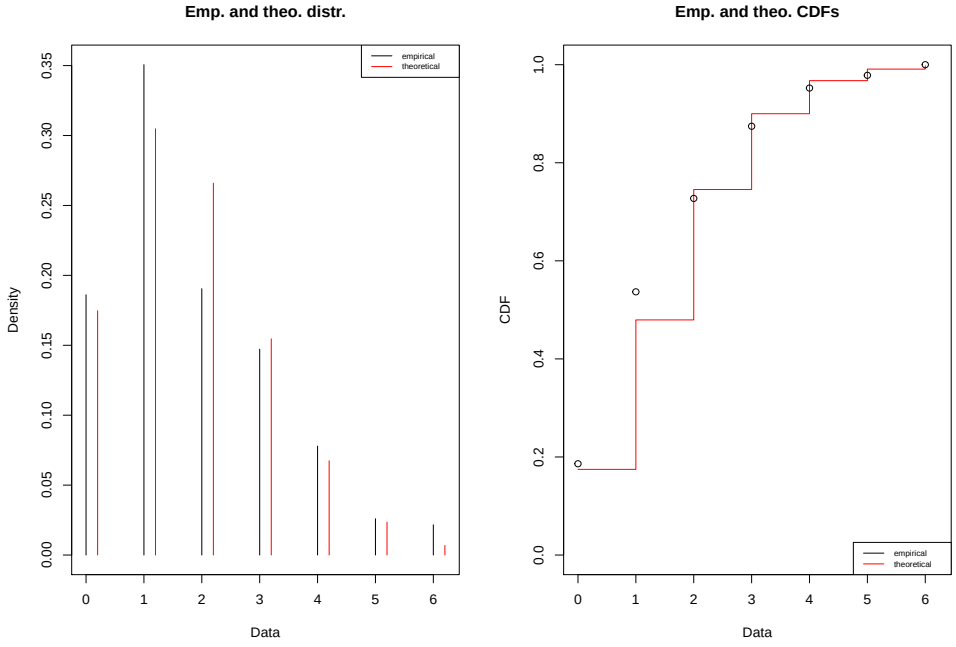


Figure 6.28: Ubuntu $n-1$ users per conversation with fitted Poisson PDF and CDF



method applied. The values computed were 2.97, 4 and 0.56 respectively. It was noted the p-value was above the 0.05 significance.

Figure 6.28 illustrates the PDF and cumulative density function CDF plots of user counts per conversation for the enterprise dataset. However, using the raw counts, a Poisson distribution was found to be a poor fit. The level of dispersion was calculated as 0.53, which indicates a moderate level of under-dispersion. A similar hurdle method was applied to the count data as described for the enterprise data set. A chi-squared test statistic, degrees of freedom and p-value were calculated with the hurdle method applied. The values computed were 11.08, 5 and 0.05 respectively. It was noted the p-value when rounding to two decimal places, was exactly 0.05 and not above the 0.05 significance.

6.4 Discussion

Prior to a detailed discussion of our results, we summarise the results along with each corresponding research question. Tables 6.3, 6.4 and 6.5 provide this summary.

6.4.1 Conversation duration modelling

The results section has shown that a parametric approach to model conversation durations is reasonable. For the enterprise dataset, a Weibull distribution was a fair fit in terms of Anderson-Darling test statistic value (1.30). However the Q-Q plot may indicate a problem with this fitted distribution, given the lack of fit at larger quantile values.

For the Ubuntu dataset, either a Burr or log-logistic distribution proved to be a reasonable fit. The computed Anderson-Darling statistic was 1.10 (for both distributions) indicates the goodness of fit. The Q-Q plot does suggest that the fit looks reasonable at small to medium quantile values. However, at large quantile values, the fitted distribution may be problematic, given the distance of the plotted quantiles from the distribution line.

Of interest is that, to produce the above fit, given that conversation duration times were measured in minutes by the system logs, durations of 0 minutes were removed from the dataset. The removal of 0's from a data set is typically undertaken as part of a hurdle model technique. We feel this is a reasonable approach as we are primarily interested in modelling conversations of a positive

Table 6.3: Summary of research question, results and techniques used

Dataset	**Research Question**	**Results**	**Data Transformation & Fitting techniques**
Enterprise	1. What method can be used to model conversation duration times?	Weibull distribution is the best fit. AD test statistic = 1.30 p-value = 0.23	Zero values removed using Hurdle method. [81] MLE [71] [72]
Ubuntu	1. What method can be used to model conversation duration times?	Burr / log-logistical distribution is the best fit. AD test statistic = 1.10 p-value = 0.31	Zero values removed using Hurdle method. [81] MLE [71] [72]
Enterprise	2. What method can be used to model conversation delta times?	(Combined) No suitable parametric fit found. (Combined) Uniform kernel with SJ-dpi bandwidth = 56.73 (Entangled) Weibull distribution is the best fit. (Entangled) AD test statistic = 0.30, p-value = 0.94 (Logical) Weibull distribution is the best fit. (Logical) AD test statistic = 0.49, p-value = 0.76	(Combined) KDE Non-Parametric method used. [82][83] [86] (Entangled) Absolute values used. MLE [71] [72] (Logical) MLE [71] [72]

Table 6.4: Summary of research question, results and techniques used (Continued)

Dataset	**Research Question**	**Results**	**Data Transformation & Fitting techniques**
Ubuntu	2. What method can be used to model conversation delta times?	(Combined) No suitable parametric fit found. (Combined) Gaussian kernel with rule-of-thumb bandwidth = 2.94 (Entangled) log-logistic distribution is the best fit. (Entangled) AD test statistic = 0.60, p-value = 0.64 (Logical) log-logistic distribution is the best fit. (Logical) AD test statistic = 2.46, p-value = 0.052	(Combined) KDE Non-Parametric method used. [82][83] [85] (Entangled) Absolute values used. MLE [71] [72] (Logical) A 1 minute value was added to each delta time (x+1). MLE [71] [72]
Enterprise	3. What method can be used to model conversation inter-arrival times?	Weibull distribution is the best fit. AD test statistic = 0.78 p-value = 0.50	MLE [71] [72]
Ubuntu	3. What method can be used to model conversation inter-arrival times?	log-logistic distribution is the best fit. AD test statistic = 0.72 p-value = 0.54	A 1 minute value was added to each inter-arrval time (x+1). MLE [71] [72]

Table 6.5: Summary of research question, results and techniques used (Continued)

Dataset	**Research Question**	**Results**	**Data Transformation & Fitting techniques**
Enterprise	4. What method can be used to model conversation message and word counts?	(Messages) Burr distribution is the best fit. (Messages) AD test statistic = 2.13, p-value = 0.08 (Words) log-logistic distribution is the best fit. (Words) AD test statistic = 0.65, p-value = 0.60	MLE [71] [72]
Ubuntu	4. What method can be used to model conversation message and word counts?	(Messages) Burr distribution is the best fit. (Messages) AD test statistic = 1.76, p-value = 0.13 (Words) Burr distribution is the best fit. (Words) AD test statistic = 0.31, p-value = 0.93	MLE [71] [72]
Enterprise	5. Can a Poisson distribution be used to model conversation user counts?	Strong evidence to suggest Poisson is a good fit. $\chi^2 = 2.97$ degrees of freedom = 4 p-value = 0.56	User counts were reduced by 1 for all values and $n-1$ users were modelled and fitted. [81]
Ubuntu	5. Can a Poisson distribution be used to model conversation user counts?	Borderline evidence to suggest Poisson is a good fit. $\chi^2 = 11.08$ degrees of freedom = 5 p-value = 0.05	User counts were reduced by 1 for all values and $n-1$ users were modelled and fitted. [81]

duration. It should be noted that the percentage of conversations removed were 23% and 17% for the enterprise and Ubuntu datasets respectively.

This study as has answered our first research question: Can the duration of annotated chat conversations be modelled by a parametric method. Data analysts from micro teams and startups can use the result of this work to compute a mean and standard deviation for their modelled distribution. These measures of location can then be used to compute the expected duration of a conversation and the proportion of conversations that will last a fixed duration. If we think of conversations within a real-time messaging application as vehicle to discuss and diagnose complex problems, this result can be used as a way to model service time diagnosis and resolution. For example, if a team regularly discusses customer issues, these chat durations can be modelled to understand whether the duration of these types of conversation is decreasing, increasing or static over time.

However, we observed that a log-normal distribution was not a reasonable fit for either the enterprise or Ubuntu dataset. This result is a little disappointing in that no direct mapping could be obtained between Outage service times and conversation durations for the enterprise dataset. Weibull and Burr / log-logistic were found to provide an adequate fit. We note that service time is a function of time to detection and time to resolution. We consider that chat duration times may be wholly or in-part, detection and resolution time. We found no evidence that Cloud outage service time distributions are aligned to conversation duration distributions.

Finally, it should be noted that for each dataset, a different distribution result was produced. As we have noted previously the Ubuntu dataset has a greater ratio of messages per hour. With a high degree of short conversations posted over a condensed period of time, it seems intuitive that a heavier tailed distribution (log-logistic) would be an appropriate fit.

6.4.2 Conversation delta time modelling

We have learned from our results that no suitable parametric method could be found to model overall conversation delta times (See Table 6.2 for details). As we used a method to differentiate between entangled and logical delta times, a two-tailed histogram was produced. We have seen that by using a non-

parametric technique such as KDE, a suitable bandwidth selector and kernel shape can be computed.

We note that small values of the bandwidth parameter h make the kernel density estimate look quite noisy, whereas larger values of the bandwidth parameter h will lead to an estimate which is too smooth in the sense that it is too biased and may not reveal structural features, like bi-modality. In both our datasets, there was a concentration of points around a central value (i.e. the mode). However, the enterprise dataset did contain a smaller mode, adjacent to the primary mode.

Using a visual inspection of the fitted density estimate over the enterprise histogram, we observe that the estimate for the enterprise dataset is reasonable, almost all points are covered with the estimated density function especially the central peak at value 0. For the enterprise data set a uniform kernel with a Sheather Jones direct plugin was found to be the most appropriate method and fit. With a computed bandwidth parameter of 56.73, we conclude that there is a high amount of smoothing to provide the plotted estimate.

The visual inspection of the Ubuntu dataset illustrates that the estimated density does not cover the histogram as well. The curve falls short of the hight of the central mode, while there are a number of small structures to the left and right that are not adequately covered by the estimated density function. The bandwidth parameter calculated as 2.94, this indicates a less smooth estimate than the enterprise dataset, even so perhaps a smaller bandwidth size may be useful in covering some of the smaller structures of the histogram not covered by the curve.

We can see the results vary depending on the dataset used. For the enterprise data set a uniform kernel with a Sheather Jones direct plugin was found to be the most appropriate method and fit. For the Ubuntu data, a Gaussian kernel using Silverman's rule-of-thumb bandwidth selector yielded the best approach.

Conversely, our study found that by dividing the conversation delta times into entangled and logical delta subsets, a parametric method can be used for data modelling. We discovered that Weibull and log-logistic distributions were the closest fits for the enterprise and Ubuntu datasets. For the enterprise dataset,

we note that the Anderson-Darling test statistic was below 1 for both the logical and entangled sub-datasets, indicating a reasonable fit. Additionally, we note that the p-values computed were above 0.75 in both cases. However, the Q-Q plot surfaced a number of issues with the fit in with both the medium and large plotted quantiles. This indicates some uncertainty around the more extreme values of the dataset being fitted to a Weibull distribution.

The results for modelling the Ubuntu dataset were also mixed. For the entangled dataset, the computed Anderson-Darling test statistic was 0.64, indicating a reasonable fit. However, the logical dataset provided an Anderson-Darling test statistic of 2.46 indicating a fair to poor fit. If we look at the graphical evidence contained within the goodness of fit graphs, we see that for the Q-Q plots both Ubuntu datasets had large quantiles that appeared off the log-logistic line. With the combination of the computed Darling test statistic and the results of the Q-Q and P-P plots, the fit for the two Ubuntu datasets is reasonable to poor, given issues with a log-logistic distribution modelling the tails of both datasets. Finally, we remark that these distributions are the same as the ones used to model conversation duration.

This piece of research has answered our second research question. For datasets with entangled and logical delta times, a non-parametric approach is our preferred option. However, if a parametric approach is required, by subsetting the data, a result showing a fair to reasonable distribution may be possible. If we think of the conversation delta times as the downtime between conversations, location measures can be computed. These measures can then be used to forecast the downtimes of team discussion in a collaborative environment. These downtime times can be used for future project planning, or personal development cycles.

6.4.3 Conversation inter-arrival time modelling

For our third research question, we asked what is the most appropriate method to model conversation inter-arrival times. We learned that once again the Weibull and log-logistic distributions were the most appropriate fits for the enterprise and Ubuntu datasets respectively. We know that the inter-arrival time is a function of conversation duration and delta times. Therefore it's intuitive that same type of distribution was found to model a time period which

spans both the duration and delta times. For both data sets, we note that the Anderson-Darling test statistics were less than 1, indicating a reasonable fit.

However, the Q-Q plot for the enterprise dataset showed a slight divergence of plotted quantiles from the Weibull line for both medium and larger quantiles. However, we note that the more extreme plotted quantiles were offset from the line. Thus the fitted distribution should be treated with some caution especially when modelling the more extreme inter-arrival time values.

Furthermore, the Q-Q plot for the Ubuntu dataset showed that as the values of the plotted quantiles increased the further these points were from the log-logistic line. On balance, while the computed Anderson-Darling test statistic indicated a reasonable fit, the log-logistic distribution may have difficulty modelling the most extreme values in the dataset.

Of interest, for the Ubuntu data set a small constant (1 minute) was added to each inter-arrival time duration. Upon review of the data, a small number of inter-arrival times were found to be of 0-minute duration. This is due, most likely, to dense bursts of messages in a collective conversation thread. Rather than remove these data points a small data transformation was applied. We note that this constant effects the overall scale of the dataset rather than the underlying shape.

The result from this work as helped answer our third research question. We wanted to understand whether inter-arrival times between conversations could be modelled effectively by a parametric method. As we can see that a parametric approach is plausible. Data scientists from startups and micro teams can use this result in two ways. As we have seen conversation duration, delta, and inter-arrival time all share a common data set on a per dataset basis, we believe this to be no coincidence given that inter-arrival time is a function of duration and delta time. Furthermore, we believe that the inter-arrival time results combined with a service time result (conversation duration) could be used as part of a queuing framework to model conversation busy times on a daily basis.

However, we recall from chapter 4 that a Pareto distribution was found to be a reasonable fit to model the inter-arrival times of Cloud outages. We observed that Pareto distribution was not a good fit to model conversation

inter-arrival times for either the enterprise or Ubuntu dataset. As a result, we found no evidence that Cloud outage inter-arrival time distributions are aligned to conversation duration distributions.

6.4.4 Conversation messages & word modelling

Our fourth research question moved the focus from conversation duration to modelling its constituent parts: the words used and the number of messages in a conversation.

We determined that for the enterprise dataset a Burr (Messages) and log-logistic (Words) were the most appropriate fits. However, some caution should be added to these statements, in both cases, the Q-Q plots point to some deviation of large quantiles from their respective distribution lines. This result may cast some doubt in the ability for this distribution to model extreme observations. Additionally, the Anderson-Darling test statistic computed for the enterprise conversations messages observations was 2.13 may not be entirely suitable to model all values observed.

For the Ubuntu dataset, a Burr distribution (Messages & Words) was the most appropriate fit. For both distributions, the Q-Q plots contained a number of extreme quantile points that deviated from the quantile line, though for smaller quantiles the plotted points did intersect the quantile line. Additionally, the P-P plot showed a fair fit for the messages data set (due to the discrete nature of the computed probability points) and a reasonable fit for the words data (almost all points fitted the probability line). We also note that the Anderson-Darling test statistic computed for the Ubuntu conversations messages and word observations were 1.7 and 0.31. On balance, the Burr distribution estimated for conversation messages is a fair to poor fit given the evidence noted, while the Burr distribution is a reasonable to fair fit given the lack of quantile fit at extreme tail values.

The Burr distribution is a flexible distribution that can illustrate a wide range of distribution shapes and types, due to the three parameters that are used to define its shape. The Burr distribution was initially used in finance (to express income levels) but has grown to wider use in areas such as hydrology (flood level modelling) and reliability (failure rates of components).

While three of the four datasets were fitted with a Burr distribution, we note

Table 6.6: Ubuntu: User counts per conversation (Untransformed Hurdle adjustment)

Transformation	**0 users**	**1 user**	**2 users**	**3 users**	**4 users**	**5 users**	**6 users**	**7 users**
Untransformed	0	43	81	44	34	18	6	5
$n-1$ users	43	81	44	34	18	6	5	NA

that the enterprise words distribution was reasonably fitted with a log-logistic distribution, given that the Burr distribution is sometimes referred to as a generalised log-logistic distribution we know there is a close affinity between both of these heavy-tailed distributions. As such, this result is unsurprising.

We have shown that a parametric approach can provide a fair to reasonable result regarding fitting messages and words to a distribution. Additionally that a heavy-tailed distribution should be used as the first port of call for distribution fitting. Using this result micro teams and startups can model their conversations to determine the expected number of messages required to conduct a conversation. Additionally, this result can be used to aid future work in the area of topic analysis of chat conversations. By understanding the expected message and words counts, suitable topic clusters and top term values can be seeded from word and message distributions.

6.4.5 Conversation user count modelling

Our final research question centred around whether a suitable method can be used to model the counts of users who participate in a group chat conversation. Typically for count data with a small number of categories, a first choice is to fit a Poisson distribution. We learned that fitting a Poisson distribution to the untransformed user count data was not a good fit for either dataset. Upon more in-depth analysis we checked to determine the level of under / over-dispersion within both each data set. It was noted that in both datasets, evidence of under-dispersion was found. However, the level of under-dispersion was greater in the enterprise dataset.

To correct the under-dispersion, a hurdle method was adopted to mitigate. Rather than remove the 0 count bin from the data set, we reduced the bin count by 1 for each dataset. Table 6.6 illustrates the Ubuntu user counts

per conversation before and after the hurdle adjustment. We found that with the hurdle adjusted data set the Poisson distribution was a good fit, for the enterprise dataset with a p-value well more than the 0.05 confidence interval. However, we found that the Poisson distribution was a borderline fit for the Ubuntu data with a p-value of exactly 0.05. We remark that the hurdle adjustment, gives a better result (regarding a better fitting p-value), when the level of under-dispersion is moderate, as can be seen in the result for the enterprise dataset. When the under-dispersion rate is slight, the hurdle adjustment appears less effective, as can be seen in the result for the enterprise dataset.

While the counts of users could not be modelled directly, we feel modelling of $n-1^{\text{th}}$ users with a Poisson distribution is a valuable result. Micro teams and startups can use this result to further future research into the field of conversation analysis. By combining conversation topic and user count modelling an enhanced model could be derived. This model could help infer what conversation topics attract large numbers of users and for large group conversations, can active and passive subsets be identified?

6.5 Conclusion

The purpose of this study was to determine what parametric / non-parametric fitting techniques could be used to model data generated from real-time chat conversations using two separate datasets. We found that a "one size fits all approach" is not appropriate. Instead, a combination of approaches is required to fit such data adequately. The findings of this study support previous study specifically in the field of Internet chat discourse inter-arrival time modelling [88, 89, 90, 91, 92, 93, 94]. These prior studies have proved useful, to illustrate how the inter-arrival times of social media (real-time chat and tweets) messages can be modelled. Additionally, previous studies have shown that the inter-arrival times of Internet chat messages can be classified as heavy-tailed datasets. By using a parametric approach, such times can be modelled by a log-logistic or Weibull distribution.

This work provided in this chapter presents a broader study of social media messaging that is not limited to modelling the inter-arrival times of messages. This study illustrates that depending on the dataset the results are different

every time.

The case study provided in this chapter found that there was no direct mapping between Cloud outage inter-arrival and service distributions and that of chat conversation inter-arrival and duration distributions. We suggest that service times are a function of both time to detection and time to resolution and that the chat conversation data may not capture the entire service time duration.

An outcome for medium sized teams from which the enterprise dataset was drawn is as follows: Team can use the modelling techniques described to determine the time to discuss / diagnose incident events. Using the distributions can help determine the proportion of incident events that take a specific duration to resolve or proportions that exceed a specific time.

For micro teams and startups, they can use the same outcome as described above to determine event proportions. Additionally, as DevOps resources are required to diagnose incidents within a real-time chat, these same resources are effectively busy. Allocating busy times for individuals can help in task scheduling to help ensure that individuals are not assigned another task while engaged in incident debugging.

In our next chapter, we continue our analysis of instant message conversations, but from two additional perspectives and case studies. The first case study looks at whether group chat messages can be segmented based on inter-arrival rates, to aid topic modelling analysis. The second case study investigates whether machine learning classification algorithms can be applied to instant message posts to infer topic boundaries.

Acknowledgment

The author would like to thank William Huber for his helpful suggestion for fitting under dispersed count data to a Poisson distribution [204].

CHAPTER 7

Chat Discourse Segmentation and Boundary Identification

Collaborative chat tools and large text corpora are ubiquitous in today's world of real-time communication. As micro teams and start-ups adopt such collaboration applications, there is a need to understand the meaning of chat conversations within collaborative teams. Additionally there is a need to segment text into meaningful topics. In this chapter we consider these domains in the form of two final cases studies. In the first study, we propose a technique to segment chat conversations to increase the number of words available for text mining purposes. We address the question of whether having more words available for text mining can produce more useful information to the end user. In the second study, we consider the problem of topic segmentation as a machine learning classification one. We also address the question as to whether a machine learning algorithm can be trained to identify salutations and valedictions within multi-party real-time chat conversations. Our results show that classification algorithms can provide a reasonable degree of precision(mean F1 score: 0.58). Our techniques can help micro-teams and start-ups with limited resources to efficiently model and classify their conversations to afford a higher degree of readability and comprehension.

7.1 Introduction

We live in an information age, where consumer-based services and applications generate more text-based data. As we embrace both collaborative and social communication, we converse more often via text-based communication [205] [206]. SMEs cite multiple benefits of real-time chat rooms [207]. Such advantages include brainstorming, client conferencing, customer support and distance learning [208]. A recent survey by ReportLinker found that while e-mail is a primary source of communication for communication, group messaging application use is on the rise [209].

However, for businesses irrespective of size, using such collaborative and social means of communication, can be an overwhelming experience [210]. This is due in part to the large volumes of text-based data that are generated by such applications and services. Recent studies have shown that almost 350,000 tweets are created every minute of every day. Globally 2.5 quintillion bytes of data are produced [211]. The growth in social media messaging is not confined to tweet messages. A recent study [212] by the Harvard business school has shown that over 2.5 billion users communicate with at least one messaging app (e.g. WhatsApp, Facebook). This figure will rise to 3.6 billion users in the next few years.

However, there are a number of drawbacks to group chat applications. Often cited are the problems with continuous partial attention (i.e. routinely checking a conversation) [213] or the lack of conversation summarisation [214]. It is the latter problem that can prove challenging for users, especially if they have been away from a group chat for a period of time. Businesses face challenges in this regard, as current group chat applications offer little or no chat summarisation functionality. Therefore, for this study, we consider techniques that may help teams make sense of their message based data.

Topic modelling is a frequently used process to discover semantic structure within a text corpus. This technique is used across multiple disciplines [215] as a vehicle to grow business insights [216]. For example, if a business can mine customer feedback on a particular product or service, this information may prove valuable [217]. One of the recommendations when employing text mining/topic modelling techniques is that the more data available for analysis, the better the overall results. However, even in the age of big data, practi-

tioners may have a requirement to text mine a single conversation or small text corpus to infer meaning.

Text segmentation is a technique used to separate text into meaningful clusters. Such clusters may include sentences or topics. Previously, text segmentation research has focused on topic changes within written discourse. Such discourse contains prose text [218]. However, in recent times attention has turned to conversational discourse such as real-time chat [219]. In an age of big data, coupled with the fact that businesses are using collaboration applications more than before [220], being able to segment chat conversations by topic may prove useful in the domain of information retrieval.

In this chapter, we conduct two case studies as follows. The first is for text mining practitioners to partition their conversations using a novel technique. Such a method can provide a higher number of words (19% on average) for topic summarisation tooling. The second, we propose a method that text segmentation practitioners can use to annotate conversations with an opening (salutation) and closing (valediction) remark. The core idea of this study is to demonstrate that by manually annotating conversation boundaries, a trained machine learning classifier algorithm can identify conversation boundaries using salutations and valedictions as a conversation perimeter.

We discussed in chapter 6 that DevOps and development teams use real-time chat software to discuss and diagnose critical defects and Cloud outage event. Chapter 6 also explored modelling of conversation durations to determine if there was a suitable analogue between chat durations and defect service times. However, the dataset used in chapter 6 had a number of limitations: Firstly the message text had to be read manually and grouped into distinct conversations. Secondly, it was challenging to understand the topic of each conversation without reading the full conversation. These are two limitations, that require further investigation.

Imagine a small team with limited resources dedicating time to initially read their group chat messages and then group into distinct conversations. Further imagine the effort required to manually extract meaning from each conversation. By leveraging techniques such as text segmentation, topic modelling and text classification, teams could identify conversation boundaries. Once these boundaries are defined the message text could be further segmented to allow

a topic modeller to provide a greater number of words for an improved level of readability of important conversation keywords.

This chapter focuses on the following research questions: a) By partitioning messages based on their inter-arrival time, can a more significant number of distinct words be returned for use by topic modelling software? b) Does a higher number of words provide a level of readability that is easier for humans to comprehend? In other words, if a user reads the terms provided by topic modelling burst and reflection periods, can they determine the context of the terms easier than a set of terms topic modelled from the entire conversation? c) Can we use the results of this work to predict an optimal topic cluster size? Using the results of this study for our framework, a topic mining solution can be developed to provide an enhanced level of understanding for small message corpora. d) What are the high-frequency words and key collocations that are present in salutation & valediction messages and e) Whether four specific machine learning classification algorithms can identify salutations, valedictions and conversation body text within multi-party chat discourse.

7.2 Case study 7 - Chat discourse segmentation

Topic modelling and text mining of social media/collaboration messaging have been shown to provide insight into the subjects people discuss as part of their online communication. By segmenting instant message text in a novel way, before topic modelling, we demonstrate how a higher degree of understanding can be achieved by the results of topic model outputs.

For this study, we topic modelled three complete chat conservations from an open source Ubuntu developer IRC channel [199]. For each conversation, IRC messages were read, we noted an initial salutation, a valediction and a grouped topic discussed in-between the greeting and farewell messages. For this study, only conversations with related topic content were considered. We note that chat conversations with mixed chat messages (i.e. 'entangled chat conversations') are beyond the scope of this study and will not be considered here. We also note that the IRC data set is the same one used in our case studies presented discussed in chapter 6

Table 7.1: Summary of Dataset Conversation Metrics

Metric	Chat 1	Chat 2	Chat 3
Total Messages	46	70	59
Total Words	292	436	484
Non-Stopped Words	158	239	262
Distinct Non-Stopped words	111	168	186
% Words for analysis	38	39	38

Table 7.1 provides a summary of the number of total words, the non-stopped words, the distinct non-stopped words and the percentage of words available for analysis.

This study aims to answer the following questions. First, can we segment a chat conversation in such a way as to provide a greater number of distinct words for topic modelling algorithms? Second, if a reasonable segmentation method can be found, is the output from a topic model easier to infer meaning, then modelling the entire conversation alone? Third, is there a relationship between the topic modelling cluster size and the number of words Input/Output from topic modelling?

7.2.1 Conversation segmentation

A question for practitioners of topic modelling is, *how can we maximise the number of words for analysis?* We know from prior research that text mining algorithms may require some form of text pre-processing prior to topic modelling. Pre-processing may include at least one of the following: Tokenisation, stop word removal, stemming and lemmatisation (note: See chapter 2 section 2.7 for more details). The removal of words as part of this pre-processing step usually is not an issue for a large text corpus, due to the number of words available. In the case of small text corpora, the problem may be more acute. For our study, stemming and lemmatisation was not conducted.

We recorded the inter-arrival time of instant message posts within the Ubuntu IRC channel, and grouped messages by short and long inter-arrival times. For successive messages with a zero minute inter-arrival time, we define this collection of messages as a burst. For messages with a one minute or greater inter-arrival time, we define this group of messages as a reflection. We then

Table 7.2: Summary of Differences between Questionnaire Samples

Sample 1	Entire chat - Topic Modelled
Sample 2	Burst & Reflections - Topic Modelled
Sample 3	Entire chat - Stop words removed
Sample 4	Entire chat - No text pre-processing

perform text mining on each burst and reflection period and then aggregate the output terms. For topic text mining, we used the tool biterm, which is suited to modelling small text corpora.

7.2.2 Topic modelling comprehension

After a conversation has been a) segmented into burst and reflection periods, b) these periods topic modelled and c) the results aggregated, we consider the efficacy of the output.

We accept that the terms output from a topic model algorithm is not explicitly designed for a readable summary. Instead, they are designed to give a user an indication of the terms used in a text corpus. Nevertheless of interest is how a user can understand the output of text mining. Our approach is to prepare four sets of text as follows; 1) each conversation is modelled with biterm (as a whole) and the mined terms output into a single collection, 2) the bursts and reflections from each conversation are modelled individually, the terms are then aggregated into a single collection, 3) each conversation with the stop words removed and 4) the raw conversation (i.e. without any pre-processing). Table 7.2 summarises the level of pre-processing conducted for each sample.

We then asked twenty four test subjects to summarise each of the four text sets belonging to a single conversation. Additionally, we asked each participant to comment on which of the four text sets was easiest to summarise. Next, we asked each subject, whether they felt set one (all terms topic modelled) or set two (bursts and reflections topic modelled) was most natural to summarise. Finally, we asked each subject to describe why they felt the text set chosen in the second question was easiest to summarise. Results of a meta-study on sample sizes for qualitative studies [221] show there is variability in sample size depending on the subject domain. For our questionnaire, twenty-four

individuals were selected, and each conversation was randomly distributed among the users.

Finally we compared the readability of every text set for each conversation using a number of known readability tests such as; Dale-Chall [222], Coleman-Liau [223], Flesch-Kincaid [224] and Gunning Fog [225].

7.2.3 Term cluster size prediction

Topic modelling algorithms use a unique set of words from a corpus for analysis. Also, we know that the process of text mining may be, in part non-deterministic. In other words, random sampling is often used to generate a term list. One of the goals of text mining is to ensure that a sufficient number of words are output in each topic cluster. The intuition is that the more unique words that are provided, perhaps the easier the output will be to understand.

Biterm, outputs topic mined terms as 'topic clusters'. Each cluster has a maximum size of ten terms. If one hundred words are input for analysis, the intuition is that ten clusters will be output with a ten distinct words. However, due to the underlying random nature of the sampling algorithm used, this is not always the case. Therefore, it is necessary to use a range of cluster sizes to obtain the optimal number of terms. We define 'optimal output words' as the number of words that is closely equivalent to the number of words used for biterm analysis. We define the 'optimal # clusters', as the smallest number of clusters that contains the optimal output words.

Simple linear regression is a statistical method that allows us to summarise and study relationships between two continuous variables. One variable, denoted x, is regarded as the predictor or independent variable. The other variable, denoted y, is regarded as the response or dependent variable.

Poisson regression is a form of generalised linear model that is used when to study the relationships between two discrete variables in the form of count data. Again we have the same predictor (x) and response (y) variables as with simple linear regression.

We used the lm function found in the base R package [176] to perform a simple linear regression test and the glm function found in the base R package [226]

We will initially use simple linear regression to explore the relationship be-

tween the number of unique words input, the biterm cluster size and the individual terms output. For example, if we model the unique words input to biterm, the cluster size that provides the unique optimal set of terms and the count of these text mined terms, a linear model could be used to predict optimal term cluster size. If a suitable model cannot be found using simple linear regression, Poisson regression will be used as an alternative approach.

7.2.4 Limitations of dataset

The dataset has some practical limitations, which are now discussed. The process of aggregating chat messages into a cohesive conversation is a subjective one. Every effort was made to assign messages to their most appropriate thread. We recognise that the process is subjective. Additionally, the post times for the Ubuntu chat were measured in hours and minutes only. As a result, we defined our burst and reflection periods as timed in zero and one minute or greater duration respectively.

The chat conversations that form part of this case study are from an Ubuntu IRC developer channel. While we hope these examples will be representative of technical discussion channels, it seems unlikely they will be typical of all types of channels. Finally, the limitations mentioned here are similar to those mentioned in chapter 6, given we are using the same IRC dataset.

7.3 Results

We now explore the results of our analysis.

7.3.1 Conversation segmentation

Table 7.3 shows a summary of the topic modelling work conducted on each of the three conversations. In the first experiment, the entire discussion was mined. In the second experiment, the burst and reflections were modelled separately.

For conversation one, a total of 96 terms were output by biterm when modelling the entire text, whereas 87 and 60 terms respectively were output from the burst and reflection analysis. A total of 51 (17%) more terms were output from the combined burst and reflection analysis than modelling the entire conversation.

Table 7.3: Summary of Text Mining Analysis: Entire Conversations Vs Burst and Reflections

Metric	**Chat 1**	**Chat 2**	**Chat 3**
Total words	292	436	484
Non-stopped words	158	239	262
Distinct non-stopped words	111	168	186
Distinct non-stopped terms output	96	129	143
# Words not analysed	196	307	341
% Words for analysis	38	39	38
% Actual terms output	33	30	30
Total burst words	185	226	287
Non-stopped burst words	98	143	163
Distinct non-stopped burst words	91	118	154
Distinct non-stopped terms output	87	118	145
# Burst words not analysed	94	108	142
# Bursts	7	11	8
% Words for analysis	49	52	54
% Actual terms output	47	52	51
Total Reflection words	107	210	197
Non-stopped reflection words	61	99	99
Distinct non-stopped reflection words	60	95	94
Distinct non-stopped terms output	60	93	94
# Reflection words not analysed	47	115	103
# Reflections	7	12	9
% Words for analysis	56	45	48
% Actual terms output	56	44	48

For conversation two, a total of 129 terms were output by biterm, whereas 118 and 93 terms respectively were output from the burst and reflection analysis. A total of 82 (19%) more terms were output from the combined burst and reflection analysis than modelling the entire conversation.

For conversation three, a total of 143 terms were output by biterm, whereas 145 and 94 terms respectively were output from the burst and reflection topic mining. A total of 96 (20%) more terms were output from the combined burst and reflection analysis than modelling the entire conversation.

In the third experiment, the stop words were removed from each of the three

Table 7.4: Summary of Text Sample Questionnaire Answers (Q1 & Q2)

Question	**Sample 1 - biterm (All text)**	**Sample 2 - biterm (Burst & Reflections)**	**Sample 3 - (Stop words removed)**	**Sample 4 - (Full text)**
One: Of the 4 text samples, which sample did you find easier to summarise? (1/2/3 or 4)	0	0	2	22
Two: Of samples 1 and 2, which sample did you find easier to summarise? (1 or 2)	0	24	NA	NA

chat conversations, no segmentation was conducted.

7.3.2 Topic modelling comprehension

Recalling the survey questions asked: Of the four text samples, which sample did you find easier to summarise? And of samples 1 and 2, which sample did you find easier to summarise? Table 7.4 shows a summary of the answers to the questions asked of the test subjects. Before the questionnaire, the subjects were asked to summarise the four samples. The questions were asked directly after the summarisation task. As we can see for question one, the majority of users found sample 4 easiest to summarise. For question two, the respondents answered unanimously in favour of sample 2.

Question three asked: For the sample, you chose in question two, why did you find that text sample easier to summarise? Figure 7.1 shows a word cloud generated from the answers respondents provided. When stop words were removed, the following terms appeared most frequently: easier (8 times), text (6), words (5) and flow/natural/understand (all 5).

To further understand the readability of text output from our topic mining experiments, we conducted some readability tests (Dale-Chall, Coleman-Liau,

Figure 7.1: Word Cloud of answers from survey question 3

Flesch-Kincaid and Gunning Fog) against each of the four text samples for all three conversations. Figure 7.2 shows a bar chart of mean readability index scores. In all cases, a lower score indicates a more readable text sample. Intuitively we can see that text sample 1 had the highest score across all indices, and text sample 4 had a lowest. Text sample 2 had a lower index score than sample 1 in all readability tests.

7.3.3 Term cluster size prediction

Our third research question asked, "Can we use the results of our topic modelling to predict an optimal topic bundle size?" We mentioned previously that obtaining an optimal number of terms (i.e. an output number of distinct words that matches an input number of distinct words) from biterm is an iterative process.

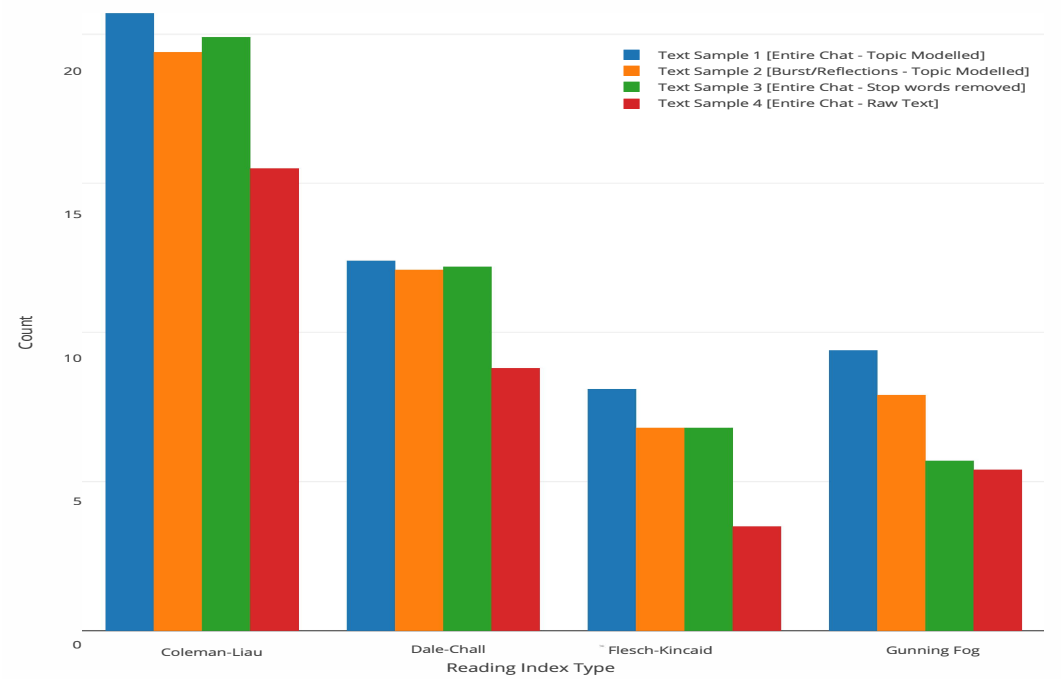


Figure 7.2: Mean readability index score of each text sample

For each burst, reflection and complete segment we topic modelled multiple cluster numbers to obtain the optimal number of distinct words. Once an optimal cluster size was found, the number of clusters was noted. We then conducted a simple linear regression test to explore the relationship between the number of distinct words output and the cluster size. In order words can we use a linear function to predict the number of topic clusters, if the optimal number of terms are known?

Figure 7.3 shows the four goodness-of-fit plots generated from our simple linear regression model. Figure 7.4 shows a histogram of the model standardised residuals plotted. We use these plots to determine the suitability of a simple linear model.

Looking at this plots in more details we note the following points. Firstly the histogram shows evidence of skew, ideally the standardised residuals should be normally distributed for simple linear regression. Secondly, there are a number of points in the Q-Q plot that lie outside of the normal line. Points 1, 16 and 41 are identified. Next, looking at the residuals Vs leverage plot

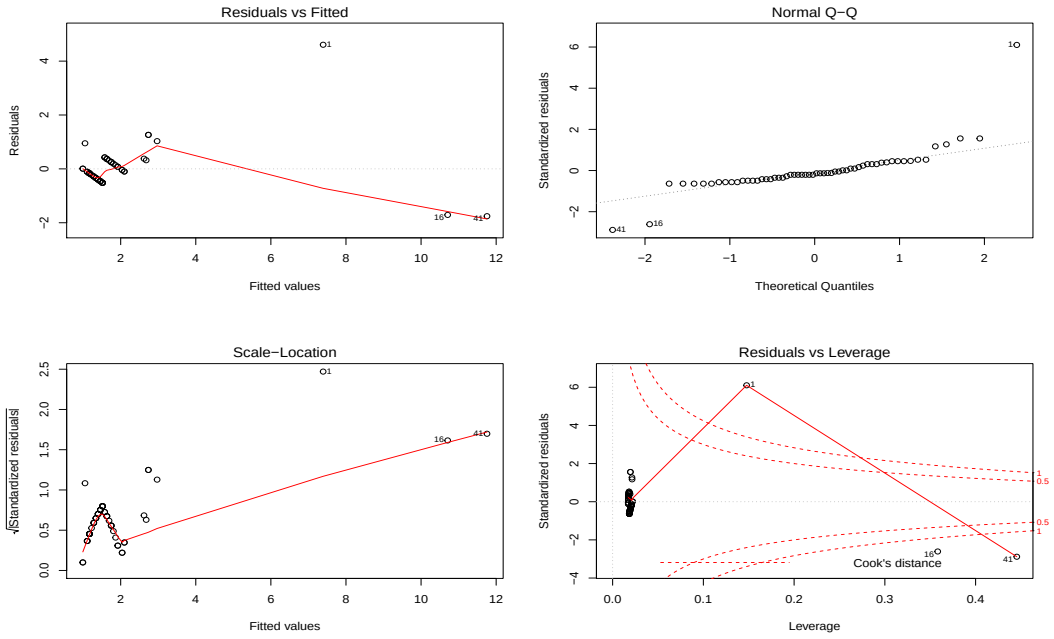


Figure 7.3: Residuals Vs Fitted, Q-Q Plot, Scale-Location & Residuals Vs Leverage for Simple Regression Model

Table 7.5: Simple Linear Regression Coefficient Table

Coefficients	**Estimate**	**Std. Error**	**t value**	**Pr(>t)**
(Intercept)	0.934	0.122	7.606	3.45e-10
Optimal.Terms	0.058	0.003	18.201	<2e-16

next, we see that these three points have a leverage greater than 0.1. This indicates these points are potentially extreme values, and the regression model is unduly influenced by them.

Table 7.5 shows the output of the Linear regression analysis. We also note the following additional outputs; Residual standard error: 0.818, Multiple R-squared: 0.855, Adjusted R-squared: 0.853 and p-value: <2.2e-16. From the output we can see that the equation to fit our linear model is as follows:

$$NumberofClusters = 0.934 + 0.058(Optimal.Terms) \tag{7.1}$$

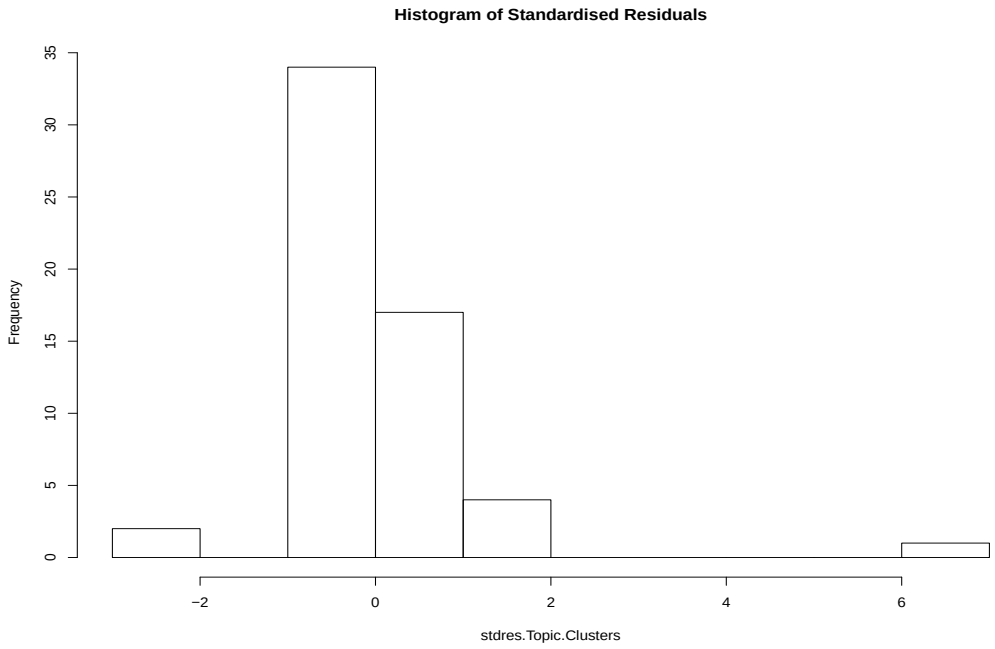


Figure 7.4: Histogram of Model Standardised Residuals for Simple Regression Model

Figure 7.5 shows a scatter plot of both the optimal terms and topic clusters variables with a fitted simple linear regression line.

Figure 7.6 shows the four goodness-of-fit plots generated from our Poisson regression model. Figure 7.7 shows a histogram of the model standardised residuals plotted. We use these plots to determine the suitability of a Poisson model. Looking at this plots in detail the we note the following points of interest. Firstly the histogram shows slight evidence of skew, while the histogram is not symmetrical it is an improvement over the simple linear model. That said the frequency of the standardised residuals are greatest at -0.5 and 0.5.

There are a number of points in the Q-Q plot that lie outside of the normal line. Points 1, 18 and 38 are identified. Next, looking at the residuals Vs leverage plot next, we see that point 38 has the largest standardised residual value (just under 2.0), while point 1 has a leverage value greater than 0.15 and a standardised residual greater than 1.5 and point 16 has a leverage of approximately 0.3. Finally, looking at the Residuals Vs Fitted graph, we can see variability in the plotted values, a slight 'saw-like' pattern is noticeable.

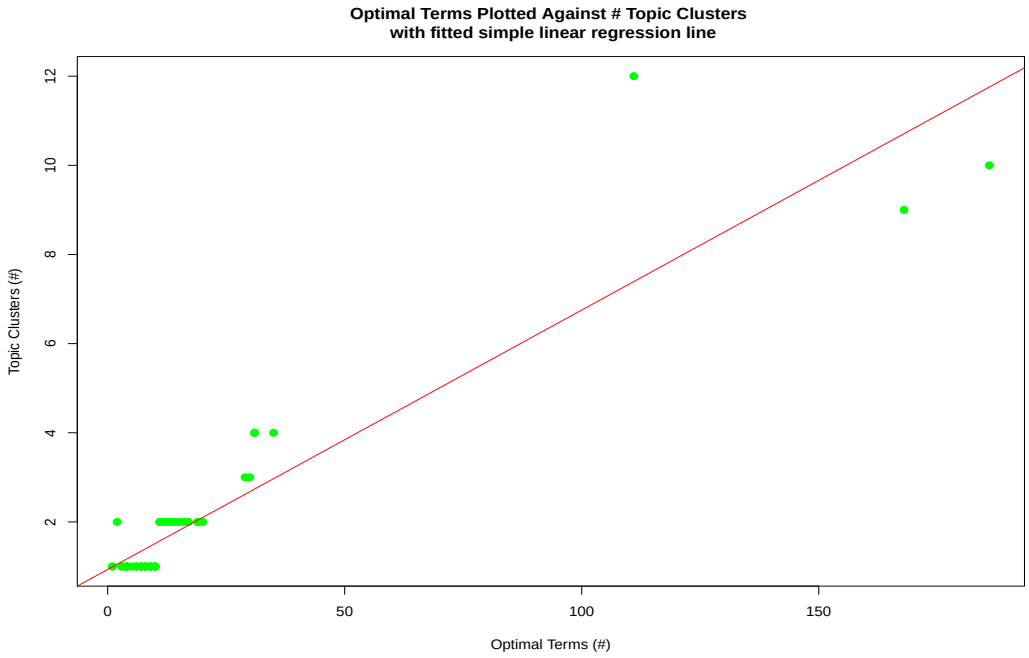


Figure 7.5: Optimal Terms Plotted Against Topic Clusters with fitted simple linear regression line

Table 7.6: Poisson Regression Coefficient Table

Coefficients	**Estimate**	**Std. Error**	**t value**	**Pr(>t)**
(Intercept)	-1.0408	0.2482	-4.193	2.75e-05
Optimal.Terms	0.6522	0.0740	8.814	<2e-16

Table 7.6 shows the output of the Linear regression analysis. We also note the following additional outputs; Null deviance: 77.395 on 57 degrees of freedom & Residual deviance: 10.073 on 56 degrees of freedom. From the output we can see that the equation to fit our Poisson regression model is as follows:

$$log(NumberofClusters) = -1.0408 + 0.6522(Optimal.Terms) \qquad (7.2)$$

Figure 7.8 shows a scatter plot of both the optimal terms and topic clusters variables with a fitted Poisson regression curve.

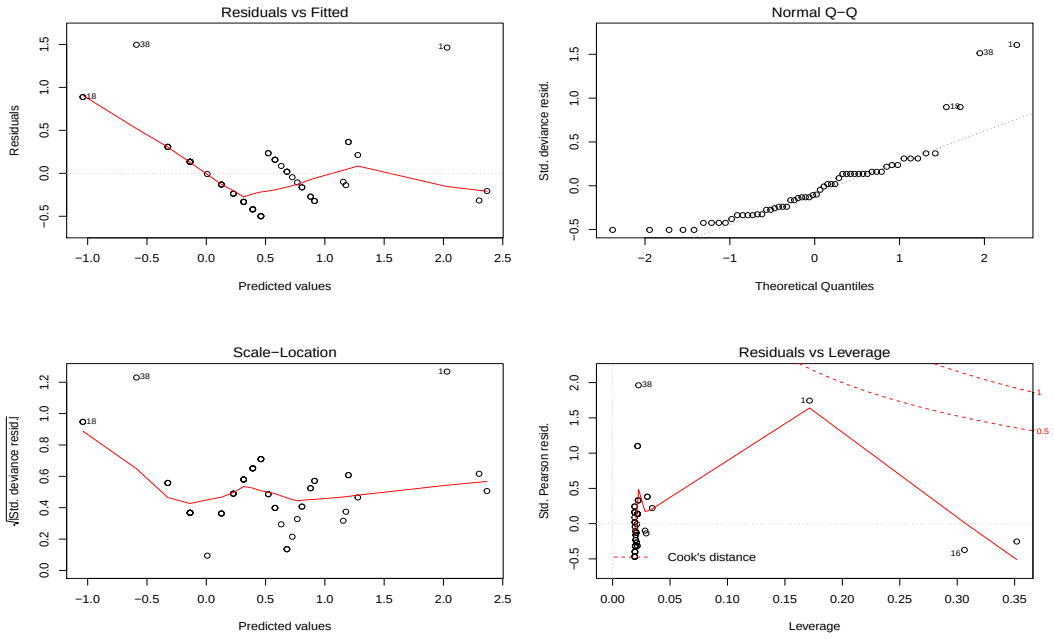


Figure 7.6: Residuals Vs Fitted, Q-Q Plot, Scale-Location & Residuals Vs Leverage for Poisson Regression Model

7.4 Discussion

The following section provides deeper analysis and discussion of the results. In each section, references will be made to each research question asked in section 7.2.

7.4.1 Conversation segmentation

Our first research question asked, can we segment a chat conversation in such a way as to provide a higher number of unique words for topic modelling algorithms? Table 7.3 shows that the mean proportion of words available for analysis for topic modelling of an entire conversation is 38%, this is due to the considerable number of stop words that are removed as part of pre-processing. Likewise, the mean number of terms output from biterm is 31% a reduction of 7%.

Conversely, when both burst and reflections are aggregated, the mean proportion of terms available for analysis is 51%. Furthermore, the mean proportion of terms output from biterm is on average, 50% a reduction of only 1%.

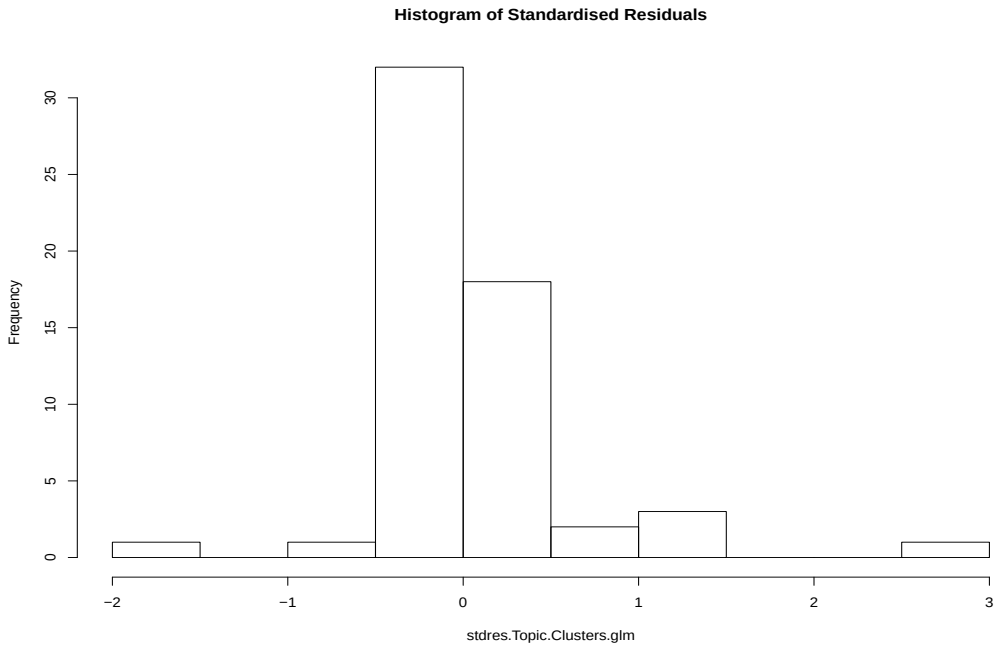


Figure 7.7: Histogram of Model Standardised Residuals for Poisson Regression Model

There is evidence to suggest that segmenting conversations into shorter segments provides a greater number of words for topic analysis due to the lack of duplicate words found in each smaller segment. We note that stop words are removed irrespective of the segment size.

Some interesting points are raised by our analysis. When stop words are removed, duplicate occurrences of the same word are also discarded. However, in the case of a more substantial text corpus, some duplicate non-stop words remain, these words are ignored by text mining tools. We see this is not the case with burst and reflection text segments. In fact, for conversation 2, 143 non-stopped words were retained. A further 25 non-stop duplicate words were ignored. In all other cases, the number of duplicate words ignored by biterm after stop words were removed was less than 10.

Furthermore, we observed that the number of burst and reflections created might have little significance on the number of terms output. Conversations 1 & 3 had a similar number of segments (i.e. between 7 and 9), while conversation 2 had 11 burst and 12 reflections respectively. While no formal

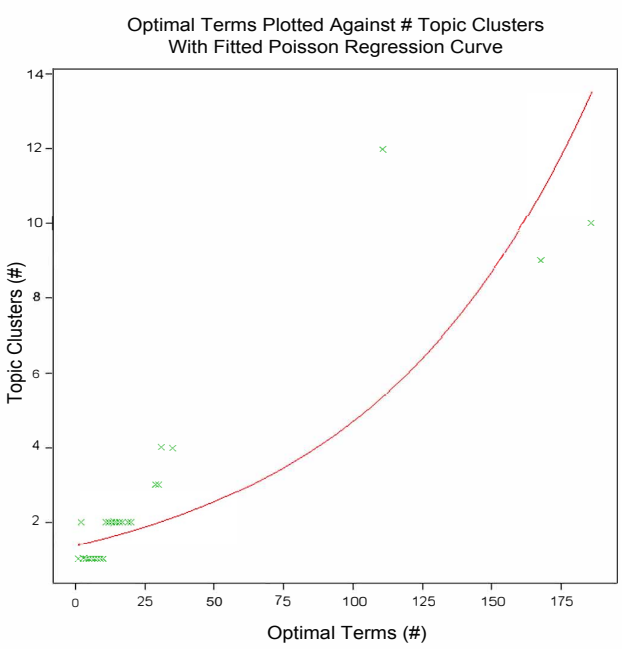


Figure 7.8: Optimal Terms Plotted Against Topic Clusters with fitted Poisson regression curve

correlation tests were conducted, when we look at the segment size and the % terms output, there seems little positive or negative relationship between the two variables.

7.4.2 Topic modelling comprehension

Our second research question asked: If a reasonable segmentation method can be found, is the output from a topic model easier to understand, than modelling the entire conversation alone? In other words, even if more words can be output as part of our improved segmentation technique, how does this translate into comprehension by both a human and for general readability.

Table 7.4 summarises the answers to the first two survey questions asked by the 24 individuals who took part in our topic modelling comprehension experiment. Unsurprisingly we can see that the majority of respondents picked sample 4, as the easiest to summarise. The consensus was that with all words available and with grammar respected (to a degree) sample 4 was easiest to summarise for the majority. However, we note that two respondents picked

sample 3 (stop words removed). The feedback from these two participants was that the samples with fewer words were easier to understand, this may be because these two individuals were not Ubuntu experts.

For survey question 2 the unanimous feedback from all users was that sample 2 was much easier to read than sample 1. A word cloud produced from the short answers provided, clearly indicate that a combination of our segmentation technique and biterm preserved the natural flow of the conversation to the degree that it was easier to summarise the text sample than sample 1.

Turning to the readability tests conducted, we can see that sample 4 produced the lowest mean index score indicating that the unprocessed chat was the most readable based on the four tests conducted. Except for the Gunning Fog index, sample 2 had equal or lower readability scores than sample 3.

It would be over-simplistic to state that when more words are available, it is easier for a human to understand a text segment based on a list of topic terms. However, it seems reasonable to assert that when more words are available and when words are placed in a similar order as to how they were typed, it is easier for humans to comprehend. What was interesting to note for short burst and reflection segments, (i.e. ten words or less) the input order of words was the same as the output terms produced by biterm. That is to say the word at the start of the sentence had the highest log-likelihood value, while the word at the end of the sentence had the lowest log-likelihood value.

We note that the goal of this research question was not to provide a readable summary based on text mined terms. Instead, the goal was to assess the understanding of text samples to humans when varying degrees of topic mining is conducted.

7.4.3 Term cluster size prediction

Our third research question asked: Can we use the results of our topic modelling experiments to predict the optimal cluster size? Previously, we discussed the problem of determining the number of clusters that will return the highest number of distinct words from the biterm analysis. We also mentioned that the optimal cluster number could be obtained only by iteratively trying a range of sizes.

Table 7.5 provides the output of a simple linear regression experiment whereby we used the optimal terms to predict cluster size. The first point to note is that the p-value for optimal.terms was <2e-16, this figure indicates a strong correlation. Additionally, we note that the multiple R-squared and adjusted R-squared values were 0.855 and 0.853. These values indicate the model contains 85% of the variability of the response data around its mean.

Figure 7.3 depicts four goodness-of-fit plots to assess the goodness of fit of our model graphically. The residuals Vs fitted plot shows our model passes through the majority of fitted values quite well. It appears that a small number of points are outside the fitted line. The normal Q-Q plot shows the standardised residuals fitted against a normal distribution line. For the most part, almost all values fit the line. However, there are three values outside of the normal line. Additionally these same three values have a leverage greater than 0.1, indicating extreme values. Figure 7.4 indicates a lack of normality of the standardised residuals. These findings cast some doubt on the suitability of a simple linear model.

Figure 7.5 shows a scatter plot of optimal terms and topic clusters with fitted simple linear regression line. The line passes through small values of topic cluster and optimal terms. As the values of both sets of variables increase the line is more distant from the observed values.

Table 7.6 provides the output of the Poisson regression experiment whereby we used the optimal terms to predict cluster size. Similar to the simple linear regression model, the p-value for optimal.terms was <2e-16, this figure indicates a strong correlation.

Figures 7.6 & 7.7 show the diagnostic plots for the Poisson regression experiment. There are similar issues with this model compared to the simple linear model, in that there are a number of points that have been flagged with either high leverage or large standardised residuals. The Q-Q plot shows these same points positioned off the normal line. That said the degree to which these point lie away from the normal line is significantly less than that of the simple linear model.

The histogram of the standardised residuals show a slight lack of symmetry, with slightly more positive than negative residuals. Finally, we note the de-

viance dropped from 77.395 to 10.073 with a loss of one degree of freedom. Deviance is used as a measure of goodness-of-fit. The reduction illustrates a greater level of fit when we use include the Optimal.Terms variable (Residual Deviance value) than using the intercept along (Null Deviance Value). This metric provides additional evidence of a reasonable fit of the Poisson regression model.

Figure 7.8 shows a scatter plot of optimal terms and topic clusters with fitted Poisson curve. We note the fit is fair. The curve appears to pass close to small values of optimal terms and topic clusters, however as the optimal terms and cluster size increases, the points appear more distant from the curve line.

We mentioned previously that biterm topic clusters contain ten terms, and that due to random variation of the tooling, it is not always possible to obtain the same level of distinct output terms as were input. For example, dividing the number of distinct words for analysis by ten and using this result as the optimal # of clusters, does not provide the same amount of output words. However, we did not know to what degree of fit a regression model would provide. In our case, a fair fit was obtained with a Poisson model. The simple linear model was not considered further due to an observed lack of normal standardised residuals.

The main benefit of such a Poisson regression model is as follows: Determining the optimal cluster size can be a time-consuming task, especially for large datasets. Even using a rule of thumb such as a 'divide input distinct words by ten' as a starting point, multiple iterations of biterm may be required. By using a linear model, the task of determining the optimal term cluster size may be expedited. In the case where the optimal cluster size needs to be computed at scale, a linear model may be more effective than iteratively computing the size using specialised hardware.

7.5 Case study 8 - Conversation Boundary Identification

Text segmentation of social media/collaboration messaging can be a useful technique to improve the quality of text mining and summarisation tasks. By annotating conversation boundaries by their salutation and valedictions, we

Table 7.7: Summary of dataset metrics and boundaries

Dataset	**Enterprise**	**Ubuntudev-IRC**
Total # Messages	3261	4223
Total dataset duration (hours)	4822	86
Multi-line conversations annotated	257	207
Salutations	257	207
Valedictions	257	207

demonstrate how machine learning algorithms can be trained to identify such opening and closing remarks with a reasonable degree of precision.

We note that in case study seven we conducted topic modelling on three distinct chat conversations. However, the process of segmenting these conversations was a manual (and time-consuming task). We, therefore, explore a machine learning classification technique to annotate the perimeter of a chat conversation via its constituent parts: A salutation (greeting message) and a valediction (farewell message).

The study presented in this case study examines four hundred and sixty-four manually annotated real-time chat conversations from two datasets. The details are summarised in Table 7.7.

For each message, we noted whether it was a salutation (i.e. an opening remark, greeting etc.). Each subsequent message was read until a valediction (i.e. a closing remark, farewell or acknowledgement message) was found. With a conversation perimeter identified we assigned a numeric topic ID. A number of single-line topics were found as part of the annotation process. For this study, only multi-line topics (i.e. conversations with one distinct salutation and valediction) are considered as part of our analysis.

The first dataset analysed was from an enterprise instant message chat system which discussed cloud infrastructure problems. For our study, we reviewed approximately 3200 messages. As part of the review phase, we annotated 257 distinct conversations. The total time period analysed was approximately 4820 hours.

The second dataset[1]analysed was the open source Ubuntu dev IRC channel [199]. For our study, we reviewed approximately 4200 messages. As part of the review phase, we annotated 207 unique conversations. The total time period analysed was approximately 86 hours. This is the same data set used to model chat conversation duration and delta times (see chapter 6 for more details) and the topic modelling work presented in case study seven.

This study aims to answer the following questions. First, what types of words and collocations are contained within salutation and valediction messages? Second, can a classifier algorithm be trained to identify salutary and valedictory text from real-time chat messages, thus identify a conversation boundary?

7.5.1 Lexicography

Lexicography is the study of vocabulary meaning and its use. In recent times, research has expanded to include a corpus based approach [227]. The benefit of a corpus based approach is as follows:

- What is the frequency of word usage?
- What is the frequency of word usage across multiple senses?
- Do words have a systematic association with other words?

The advantage of the corpus-linguistic method is that language researchers can analyse naturally occurring language text produced by a variety of authors to confirm or refute intuitions about specific language features using empirical data.

We aggregated both the salutation and valediction messages from each dataset into a single corpus. We then used the corpus linguistic tool #lancsbox [228] to analyse our salutation and valediction words. Our first research question asked a) what are the high-frequency words used and b) what interesting collocations are present in salutation & valediction messages.

7.5.2 Chat boundary classification

Our second research question asked, can a machine learning classifier algorithm be trained to identify text as a salutary and valedictory text from real-time

[1]A copy of the annotated open-source dataset is available upon request.

Table 7.8: Condensed IRC conversation with classification labels

Date	User	Test	Label
01/10/04 06:20	<m_tthew>	fabbione: ahoy	salutation
01/10/04 06:27	<fabbione>	hey m_tthew	message-body
01/10/04 07:04	<mdz>	morning	message-body
01/10/04 07:11	<fabbione>	mdz: for the ati / flrdkjdjds driver...	message-body
01/10/04 07:18	<fabbione>	building now :-)	message-body
01/10/04 07:18	<fabbione>	brb	valediction

chat messages with a reasonable degree of precision. For this research question, we constructed a multinomial classification experiment using three classes; salutation (opening message), message-body (neither an opening or closing message) and valediction (closing message). Table 7.8 provides an overview of the manual classification methodology.

There is an open question as to whether stop words (i.e. the most common words in a language) should be removed before classification. If stopwords are removed, it can remove "noise" from sentences and allow a classifier to focus on a subset of text. However, the concern is that valuable text markers may be lost if such text is removed. We conduct our classification experiment on both the full text and with stop words removed.

Next, we conducted the following steps to train four classifier algorithms (Decision trees, NB, random forest and SVM) and evaluate each training set using the python library scikit-learn[2].

- Each dataset was divided into a training, development and test set, in a ratio of 60% / 20% / 20%.
- Each training set was trained against both classifier algorithms.
- A development set was used to assess the performance of each algorithm.
- A test set was used to assess the performance of each algorithm to assess over/under-fitting.

[2]http://scikit-learn.org/stable/

- The above steps were repeated with the stop words removed from both datasets.

Note: The decision tree algorithm in scikit has many configurable options. We assessed a total twenty combinations of criterion, depth, sample split, sample leaf and weight fraction as part of our initial tuning of the dev set. The random forest also has many configurable options. We assessed a total of twenty-two combinations of estimators, criterion, depth, sample split, sample leaf and weight fraction. For SVM we evaluated a total of thirty-six combinations of loss and penalty functions using the development set. The highest performing combination of options was then used to validate the training set, using the same parameters. Finally, The NB algorithm has no tuning parameters.

7.5.3 Limitations of dataset

The dataset has some practical limitations, which are now discussed. The process of aggregating chat messages into a cohesive conversation is a subjective one. Every effort was made to assign messages to their most appropriate thread. We recognise that the process is subjective, and may be subject to type I errors.

The chat conversations that form part of this study are from an Ubuntu IRC developer channel and an Enterprise Cloud channel. While we hope these examples will be representative of technical discussion channels, it seems unlikely they will be typical of all types of channels. Finally, as this is the same data set used in case studies six and seven, we note the how the limitations of the data set may impact multiple areas of this work.

7.5.4 Results - Lexicography

We tabulated a list of the fifty most common words found in both salutation and valediction text. Thirty-two of these words were found to be either stop words or usernames. We removed these words from our list. Table 7.9 provides a summary of the most common words used across the salutation and valediction corpus. We include the absolute rank of the word frequency in the overall corpus.

We choose two words: *morning* and *thanks* from Table 7.9 to explore collocations in more detail. We selected *thanks* as we observed that it was used

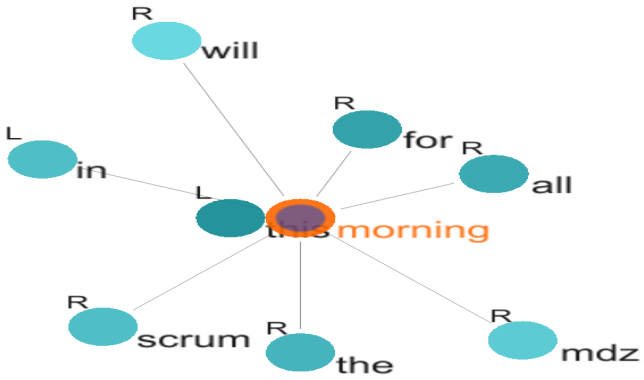


Figure 7.9: Collocation plot for the word 'morning'

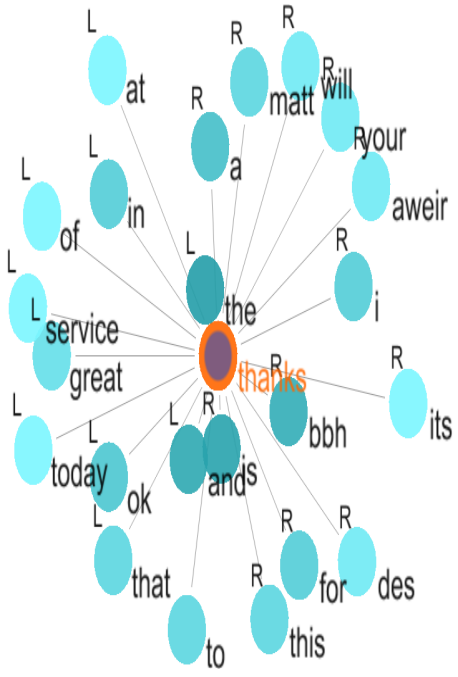


Figure 7.10: Collocation plot for the word 'thanks'

Table 7.9: List of the most frequent non stop words

#	**Word**	**Frequency**	#	**Word**	**Frequency**
17	thanks	68	35	jenkins	36
20	can	64	36	ok	36
21	now	51	37	update	35
24	will	49	40	get	35
27	today	42	42	please	33
29	not	40	43	need	33
30	as	40	44	morning	33
31	new	38	45	working	31
34	just	36	48	still	29

frequently in a number of closing messages to acknowledge completion of some task. We selected *morning* as it appeared in a number of salutations as a form of greeting. Figure 7.9 & Figure 7.10 illustrate a collocation graph for each word. A collocation graph is used to show how a given word is used in conjunction with other words, and whether that word appears to the left or right of a collocated word.

7.5.5 Results - Chat boundary classification

Table 7.10 provides a summary of the highest scores achieved from the six classification experiments conducted. We note that highest mean precision, recall and F1 score was achieved with the Ubuntu IRC dataset with stop words removed. The lowest mean precision, recall and F1 score were against the combined Enterprise & Ubuntu IRC dataset with stop words removed. For SVM the Huber loss function with no penalty provided the highest mean scores. We also note that neither the decision tree nor the random forest classifier performed as well as either NB or SVM classifiers in the six experiments conducted.

Figure 7.11 shows a plot of mean precision, recall and F1 for all twenty-four classification classification experiments conducted. Random forest achieved the twelvth highest classification score with precision, recall and F1 score of 0.51, 0.49 and 0.49 respectively. Decision tree achieved the sixteenth highest classification score with precision, recall and F1 score of 0.57, 0.45 and 0.43 respectively.

Table 7.10: Summary of the Best Performing Classification Algorithms

Dataset	**Processing**	**Classifier**	**Label**	**Precision**	**Recall**	**F1**
Ubuntu IRC	None	NB	salutation	0.52	0.55	0.54
			message-body	0.57	0.8	0.67
			valediction	0.55	0.3	0.39
			mean	0.55	0.55	0.53
Ubuntu IRC	Tokenised &	SVM	salutation	0.62	0.45	0.52
	Stopwords removed		message-body	0.54	0.7	0.61
			valediction	0.62	0.6	0.61
			mean	0.59	0.58	0.58
Enterprise	None	SVM	salutation	0.55	0.62	0.58
			message-body	0.57	0.58	0.57
			valediction	0.56	0.48	0.52
			mean	0.56	0.56	0.56
Enterprise	Tokenised &	SVM	salutation	0.68	0.64	0.66
	Stopwords removed		message-body	0.49	0.58	0.53
			valediction	0.57	0.5	0.53
			mean	0.58	0.57	0.57
Combined	None	SVM	salutation	0.59	0.52	0.55
			message-body	0.54	0.58	0.56
			valediction	0.52	0.53	0.52
			mean	0.55	0.54	0.54
Combined	Tokenised &	SVM	salutation	0.57	0.57	0.57
	Stopwords removed		message-body	0.50	0.41	0.45
			valediction	0.49	0.58	0.53
			mean	0.52	0.52	0.52

7.5.6 Discussion Lexicography

Our first research question asked a) what are the high-frequency words used and b) what interesting collocations are present in salutation & valediction messages. We note that thirty-two of the top fifty words were 'stop words', as these words are the most commonly used words in the English language this is unsurprising. Table 7.9 illustrates that *thanks, can* and *now* appear fifty times or more. We note that: *Thanks* (a noun or interjection used to express gratitude), *Can* (is an auxiliary verb used with a pronoun (e.g. you)) and *Now* (an adverb used to add a time dimension to an action or statement). Interestingly, neither a variation on the standard greeting or farewell (e.g. hello, goodbye) appeared in the top one hundred words. *Hi* was the 120th most frequent word while *later* was the 292nd most frequent word.

Looking at the collocation graph in Figure 7.9 we see *morning* collocates with nine words. The most frequent collocates were *this* (left) and *for* (right). Interestingly, morning does not collocate with *good*, however it does collocate with a username *mdz* on six occasions. Figure 7.10 shows the collocation graph for *thanks*. We can see that *thanks* collocates with twenty-nine distinct words. On twenty-four occasions, *thanks* has no collocates (i.e. used as a single word message), also on thirty-four occasions, *thanks* collocates with four distinct usernames (i.e. *bbh, matt, des, aweir*).

The key takeaway from this work is to demonstrate how complex and variable written discourse is. By adopting a corpus based approach, we can understand how language is used within a specific domain. The results of corpus analysis can be used define features for use as part of a deep learning architecture.

7.5.7 Discussion Chat boundary classification

Our second research question asked, can a machine learning classifier algorithm be trained to identify text as a salutation or valediction from real-time chat messages and if so to what degree of precision? Looking at Table 7.10 and Figure 7.11, a number of points are apparent. Overall SVM performed best in five of the six experiments, the highest mean precision, recall and F1 score achieved was with the Ubuntu dataset with stop words removed. The Huber loss function with no penalty provided the highest level precision across all experiments. Random forest performed third best overall with all six mean

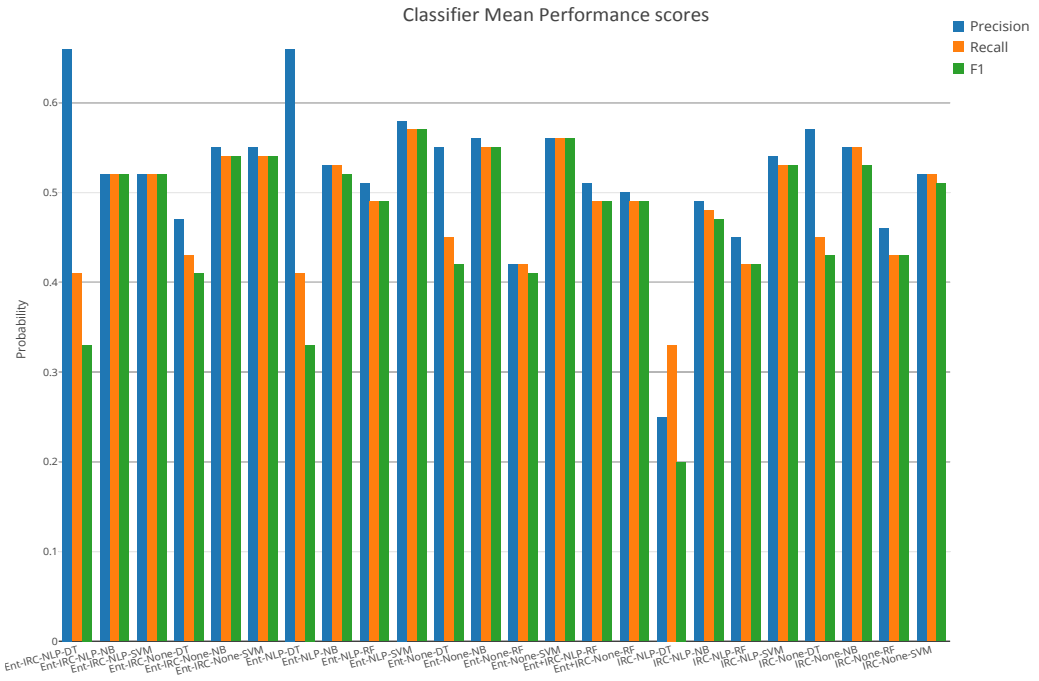


Figure 7.11: Mean Precision, Recall and F1 scores for all classification experiments.

scores between twelfth and twenty-first based. Decision tree performed the worst of all four classifiers with all six mean scores between sixteen and twenty-four.

Combining the two datasets did not yield an obvious improvement in classifier performance. By combining both datasets, we added more variance to our training and test data, which degraded classifier performance. Interestingly, when stop words were removed from each dataset, we observed a slight increase in classifier performance except with the combined dataset. Both SVM and Random forest achieved their highest mean precision scores with stop words removed. Our intuition suggests that by removing stop words, we removed some of the variance from both datasets. However, these performance gains are reduced when both datasets were combined.

Overall, SVM and NB classifier algorithms achieved average or slightly better than average performance. We see two contributing factors. Firstly, we know there is variability in the language used within salutation and valediction text. We observed that a number of conversations start with an image, URL

or emoji rather than an English word. Secondly, for the message-body label, there is even more variability due to the many ways in which humans can express themselves in chat discourse.

We also note that both SVM and NB work well on sparse datasets. Additionally, that Decision tree and Random forest work better with a mixture of numerical and categorical features, our datasets lack these features. As a result, the classification results are unsurprising.

The main benefit of such an experiment is to highlight the idea, that neither the NB nor SVM text classifier algorithms are suited to identify specific types of utterances in chat discourse with a high degree of precision. However, we acknowledge that increasing the level of training and test data would be reasonable for further experiments in this area.

7.6 Conclusion

The purpose of our last two cases studies was two-fold. First, we examined topic modelling for small text corpora (i.e. Instant message conversations). Second, we wanted to understand whether a text classification algorithm could be used to identify conversation boundaries using salutation and valediction text within a group chat context. Additionally, we adopted a corpus linguistic approach to identify lexical patterns within group chat conversations.

For case study seven, we found that by segmenting messages into periods of intense (bursts) and non-intense (reflections) communication that these segments, when used in conjunction with a text mining tool can be used to provide a higher number of output terms than modelling the entire corpus of messages at once. Furthermore, we found that the message inter-arrival time can be used to determine both burst and reflection periods.

We also found that the terms output from topic modelling bursts and reflection periods, when aggregated, is easier to understand than the text mined terms from the entire message corpus. Additionally, we saw that all four readability tests, topic modelled terms output from aggregated burst and reflection analysis have a lower readability index compared to terms mined from the entire corpus.

Also, the relationship between optimal output words and the optimal # clus-

ters was explored in detail. A Poisson regression model was found to provide a technique method to predict topic cluster size, although we note the model was a fair fit. This result can have a positive benefit for topic modelling practitioners, as it may reduce the iterative approach needed to find the number of topic clusters that produce the largest distinct number of words.

Both SMEs and micro-teams can use the above result to deliver high-value topic mining outputs from their group chat discourse. Teams can focus initially on a corpus-based approach for a particular channel/space. The advantage of a more extensive corpus approach is that topic modelled outputs can be assessed in context. Where word collocations exist, this knowledge can be directly applied to place a higher value on terms generated from topic mining tools.

For case study eight, We found that the SVM classifier provides a modest level of precision identifying opening and closing remarks within group chat conversations. Additionally, we found that classifier performance varied little between datasets and that removal of stop words increased classifier performance on each data set. Combining datasets saw a slight decrease in classifier performance. Additionally, we saw that SVM outperformed decision trees and random forest in all cases and NB in all but one experiment.

Furthermore, we found that a corpus-based approach can provide useful insights into the mechanics of opening and closing messages of a group chat conversation with the use of collocations.

By adopting a finer grained corpus-based approach, additional features may be developed that can be used to provide models with improved performance. We consider that a multi-feature classification model in the form of a neural network could be used to further our work.

In our final chapter, we provide a conclusion of the research conducted as part of this thesis. We also provide information as to how the case studies presented can be expanded as part of future work. We also offer some final closing thoughts.

CHAPTER 8

Conclusions

In this chapter, we review and summarise the work presented in this manuscript and provide suggestions for possible extensions.

8.1 Summary

The work presented in this thesis looked at three aspects of software delivery to the Cloud: Software testing, Cloud outage modelling/simulation and chat discourse modelling & segmentation.

Chapter 2 reviewed the field and the current state of related literature. Chapter 3 looked at how the customer can aid software testing in the realm of a rapid software release model. Additionally, in this chapter, we looked at the importance of internal software testing before release and proposed a method of crowdsourced testing. The Social Testing and crowdsourced contribution of this chapter can aid SMEs and micro teams to produce high-quality software while respecting their low level of resources.

Chapter 4 focused on modelling the inter-arrival and service times of Cloud outage events. By examining Cloud outage times, a suitable probability density distribution was found to model both inter-arrival and service times. We also looked at whether the distribution type varied by outage type and by software component. The core contribution to knowledge was the discovery that a Pareto distribution could be used to model outage inter-arrival times and a

log-normal distribution could be used to model outage service times. The latter result fits well with previous research that states a log-normal distribution can be used to model service times of repairable systems.

Chapter 5 takes the inter-arrival and service time distribution results from chapter 4 and uses this result to simulate Cloud outage events (in the form of busy times of a Queuing system). Furthermore we demonstrated that by using distributions modelled on outage inter-arrival and service times (i.e. a special case of a G/G/1 queue), a more accurate simulation could be achieved over an M/M/1 queue. By employing such a queue model framework, DevOps teams can build an precise planning model to ensure sufficient resources are available when a Cloud outage occurs.

Chapter 6 explored an analogue to outage service time by modelling and text mining real-time group chat conversations. By analysing two distinct datasets we found that a suitable probability density distribution could be found to model conversation duration, delta, message inter-arrival time, and message line/user/word counts. We also found that the type probability distribution is different depending on the message density. The findings of this work support previous studies in the field of chat discourse modelling.

Chapter 7 considered the area of topic modelling within small text corpora. Through analysis of multiple threaded conversations, we considered a technique for discourse segmentation, that allows a higher availability of words (between 15% to 20%) for topic modelling algorithms such as biterm and LDA. Furthermore, we demonstrated that this layering technique provides a more readable output than using an entire message corpus alone. Additionally we found it was possible to build a linear model to predict the optimal cluster size for topic modelling algorithms. The main benefit for small teams that produce large quantities of chat discourse is to aid the problem determination process by providing a higher resolution topic term outputs.

Chapter 7 also provided two additional experiments drawing on the fields of computational linguistics and document classification. We demonstrated that identification of specific regions within a conversation is a non-trivial task. Using off-the-shelf classification algorithm provides a slightly better-than-average result in detecting boundaries of chat conversations. Additionally, a corpus-based approach was used to identify collocations within opening and closing

messages. The key take away from both experiments is first to illustrate how a corpus-based method can be used to provide insight into a specific text discourse. Second, is to highlight the challenges that classification algorithms face providing a well-fitted model, especially given the many ways in which humans can express themselves in written discourse.

8.2 Future Work

Many possible additions or improvements could build on this body of work.

As we saw in chapter 3, we proposed a bounty/reward system to incentivise customers to find defects within a given product. A useful extension would be to employ a system of Gamification [229]. Gamification could be interesting here given the additional proposals of crowd-sourced testing; this would allow cohorts of testers to compete with each other to find the most bugs in a given time period. Adopting and measuring the efficacy of such a model could provide a higher degree of business intelligence to software manufacturers.

In chapter 4 we saw the benefits of how modelling inter-arrival and service times can be used to infer arrival and service remediation rates in general. With sufficient data points, SMEs and micro teams can target specific components and outage types. We acknowledge that, while a general distribution model is useful, a suite of models is of greater benefit to teams who want to first understand outage times within specific troublesome components and failure types.

An additional extension is an iterative approach to modelling failure times. For example, if teams want to understand fundamental questions such as: Are in the inter-arrival/service times of specific outages categories increasing or decreasing on a per release basis? Re-modelling on a regular basis would be useful.

Chapter 5 discussed how it is possible to use inter-arrival and service time distributions as a method to seed an G/G/1 queue to predict the busy time of teams using a simple queuing model. One natural extension of this work is to apply the approach in a queuing framework with multiple servers (i.e. an G/G/c queue). This addition appears intuitive given that applications may have various components, failure types and data centre locations. Therefore

by extending the model to predict team busy times at a specific data centre location, or within a particular sub-component or indeed based on a specific type of failure is beneficial.

Teams can then be better positioned to deploy specific resources where they are most needed. Additionally, this future work may aid training and upskilling efforts within groups. Consider the scenario whereby, team busy times are known to be clustered around a common outage type or a protracted service time engagement. By surfacing this information technical leads and management can use this knowledge to investigate methods to either reduce the occurrence of frequent outage patterns and to reduce the service time to remediate parallel outages.

By using the simple queue as a starting point, future work is planned to validate the framework in the context of a complex queuing system (i.e. a queue with multiple "servers"). John [230] discusses dependencies between inter-arrival and services times within a queue system as the assumption of independence between the two times are not always valid.

Also of interest is the wear out characteristics of specific software components. Conducting a study of Cloud outages as part of a wider renewal process study we can understand the mean time to failure of a given component. With the age (uptime) of a component known, how likely is such a component to fail? Finally how often does a component need to be replaced and what role does hygienic recycling play in system stability?

Chapter 6 provided a general framework to model the duration, delta and inter-arrival of segmented chat conversations. Possible extensions to this general framework are as follows: We know that current real-time chat offerings such as Slack, Microsoft Teams and Watson Workspace, use the construct of channels or spaces to partition conversations based on topic. By adopting a hierarchal model, teams can analyse conversations at an organisation level. Additional there is value in modelling conservations whereby both topic and users are seen as the principal components of segmented conversations. Understanding expected conversation durations, for a given topic or when a specific cohort of individuals is involved, may help in the resolution of problems where real-time chat is the primary means of communication. One final area of interest is in the area of inter-arrival times of messages, and whether

messages appear in *bursts* and whether the burst times of conversations and messages could be modelled by a Markov process.

Chapter 7 demonstrated that by segmenting messages into burst and reflection pools, a larger number of words are available for topic model algorithms and that such modelling can lead to greater readability. We acknowledge that we used a simple set of criteria to define a burst and reflection. However, we know that message density is not the same in all message corpora and that imposing a one size fits all (generalised) solution is undesirable. Therefore we suggest the following extensions. Firstly by adopting a supervised machine learning approach, a system can learn and define burst and reflection periods for a specific text corpus. Another extension to aid in this regard, is to understand what types of utterances lead to defining a burst and reflection period and to what extent subsequent messages are depending on an initial message. Finally, by using a larger set of conversations including both small and medium to large text corpora, we can understand whether a linear regression model is a use method to predict the optimal number of topic clusters, or if a Poisson model is more appropriate (given we are dealing with count data).

Chapter 7 also demonstrated that classification of text discourse is challenging, due to the many ways in which users can express themselves. We acknowledge that the boundary definitions were simplistic: The first and last message of a conversation where the boundary markers. In retrospect, a boundary may involve multiple salutation and valediction messages to form 'mini-clusters'. Additionally, as part of text pre-processing, we included usernames and URL link information. We suggest two future improvements to our work. First, by annotating conversations in the form of boundary clusters may provide much-needed training data to classification algorithms. Second by replacing distinct usernames with a username tag/token, and URL's with a similar tag/token. It may be valuable to know whether is a piece of text is a username or a URL, perhaps it is less important to understand whom the user is what websites are referenced. These efforts may reduce the variability within a text corpus.

8.3 Final Thoughts

From work presented over the course of this thesis, we have shown, that to employ rapid software delivery, software providers need not compromise on

quality. By adopting a suite of techniques, we have demonstrated ways in which teams with a small number of resources could compete (regarding rapid delivery and quality) with larger corporations.

Over the duration of this research, we have seen on the surface three seemingly unrelated topics converge into this thesis. This work is related to the way both Cloud and CD is seen as a vehicle for the rapid delivery of software. Such delivery methods were seen as the preserve of large corporations in times past. However, we know what the SME and micro team has a role to play in rapid software delivery given their contribution to the global economy [231].

The idea that customers are used to help test software they have paid for may seem unorthodox. Indeed, this viewpoint would be seen as controversial 15 to 20 years ago. However, through studies of crowdsourced programs and reward systems, this idea does not seem that radical now. We would ask readers and researchers of this work to consider ways in which to challenge their own orthodoxy. By doing so, they may find solutions to problems not previously envisaged.

We live in an age of endless data, it is essential to stop and reflect on the data we produce, and how this data can be best used, to derive a higher degree of business and user intelligence. We note that machine learning (and AI) inevitably will have a greater role to play in how the data we generate is consumed. As practitioners devise the next generation of solutions based on our endless streams of data, we also need to educate both end users and machine learning practitioners of the ethical implications of our data and derived solutions. The challenges ahead will be, in part philosophical and in part technical.

Bibliography

[1] J. Martin, *Rapid application development.* Macmillan Publishing Co., Inc., 1991. xiii

[2] H. A. Simon *et al.*, "An Empirically-Based Microeconomics," *Cambridge Books*, 2009. xiii

[3] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Pearson, 2010. xiii, 37

[4] M. Loukides, *What is DevOps?* " O'Reilly Media, Inc.", 2012. xiv

[5] G. Kim, K. Behr, and G. Spafford, "The Phoenix Project: A Novel About IT," *DevOps, and Helping Your Business Win*, 2014. xiv

[6] (2017) Outage and Maintenance definition. [Online]. Available: http://bit.ly/2xloCfL xiv

[7] I. C. S. S. E. S. Committee and I.-S. S. Board, "IEEE recommended practice for software requirements specifications." Institute of Electrical and Electronics Engineers, 1998. xiv, xv

[8] S. Deshpande, "A Study Of Software Engineering Practices for Micro Teams," Ph.D. dissertation, The Ohio State University, 2011. xiv

[9] L. Kleinrock, *Queuing systems.* Wiley, 1975. xiv, 88, 103, 113

[10] D. Gross, *Fundamentals of queueing theory.* John Wiley & Sons, 2008. xiv, 103

[11] V. Sundarapandian, *Probability, Statistics and Queuing Theory.* Phi Learning, 2009. xiv

[12] (2017) SME Definition. [Online]. Available: http://bit.ly/2ufn3is xv

[13] A. Croll and B. Yoskovitz, *Lean analytics: Use data to build a better startup faster*. "O'Reilly Media, Inc.", 2013. xv

[14] J. Dempsey, M. G. W. Davis, A. Crossfield, and W. C. Williams, "Program management in design and development," SAE Technical Paper, Tech. Rep., 1964. xv

[15] H. D. Benington, "Production of Large Computer Programs," *Annals of the History of Computing*, vol. 5, no. 4, pp. 350–361, 1983. xv, 13

[16] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proceedings of the 9th international conference on Software Engineering*. IEEE Computer Society Press, 1987, pp. 328–338. xv, 13

[17] (2015) Why Multi-Tenancy is Key to Successful and Sustainable Software-as-a-Service (SaaS). [Online]. Available: http://bit.ly/2G18J1Q 8, 60

[18] (2015) From Google to Amazon - the rise of the cloud catalog. [Online]. Available: http://bit.ly/2IsjHMn 8

[19] (2015) Pole Position: Ranking the Top 5 IaaS, PaaS and Private Cloud Providers. [Online]. Available: http://bit.ly/1UQCaSf 8

[20] (2016) Best Platform as a Service (PaaS). [Online]. Available: http://bit.ly/2bavsb5 9

[21] (2015) The 10 worst cloud outages. [Online]. Available: http://bit.ly/1ISiawO 9, 74, 86, 89

[22] (2013) Dropbox Outage Represents First Major Cloud Outage of 2013. [Online]. Available: http://bit.ly/2bjFdla 9, 74

[23] (2013) Dropbox Currently Experiencing Widespread Service Outage. [Online]. Available: http://tcrn.ch/2bDEyM5 9, 74

[24] (2015) The 10 Biggest Cloud Outages Of 2015. [Online]. Available: http://bit.ly/2ItPTPE 9, 74

[25] (2016) The 10 Biggest Cloud Outages Of 2016. [Online]. Available: http://bit.ly/2bjsPGL 9, 74, 86

[26] D. Yuan, Y. Luo, X. Zhuang, G. R. Rodrigues, X. Zhao, Y. Zhang, P. U. Jain, and M. Stumm, "Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 249–265. 9

[27] S. Hagen, M. Seibold, and A. Kemper, "Efficient verification of IT change operations or: How we could have prevented Amazon's Cloud outage," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 368–376. 11

[28] Z. Li, M. Liang, L. O'Brien, and H. Zhang, "The Cloud's Cloudy Moment: A Systematic Survey of Public Cloud Service Outage," *arXiv preprint arXiv:1312.6485*, 2013. 11

[29] M. Sedaghat, E. Wadbro, J. Wilkes, S. De Luna, O. Seleznjev, and E. Elmroth, "Die-Hard: Reliable Scheduling to Survive Correlated Failures in Cloud Data Centers," 2015. 11

[30] R. Potharaju and N. Jain, "When the network crumbles: An empirical study of Cloud network failures and their impact on services," in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 15. 11

[31] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving Failures in Bandwidth-Constrained Datacenters," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 431–442. 11

[32] M. Carcary, E. Doherty, and G. Conway, "The Adoption of Cloud Computing by Irish SMEs–an Exploratory Study," *Electronic Journal Information Systems Evaluation Volume*, vol. 17, no. 1, 2014. 11, 60, 88

[33] B. Snyder, J. Ringenberg, R. Green, V. Devabhaktuni, and M. Alam, "Evaluation and design of highly reliable and highly utilized Cloud com-

puting systems," *Journal of Cloud Computing*, vol. 4, no. 1, p. 1, 2015. 12, 84

[34] G. Q. Kenny, "Estimating defects in commercial software during operational use," *IEEE Transactions on Reliability*, vol. 42, no. 1, pp. 107–115, 1993. 12, 84

[35] P. O'Connor and A. Kleyner, *Practical Reliability Engineering*. John Wiley & Sons, 2011. 12, 84, 105

[36] R. Almog, "A study of the application of the lognormal distribution to corrective maintenance repair time," Ph.D. dissertation, Monterey, California. Naval Postgraduate School, 1979. 12, 84

[37] A. Adedigba, "Statistical Distributions for Service Times," Ph.D. dissertation, Citeseer, 2005. 13, 84

[38] R. Alsoghayer and K. Djemame, "Resource failures risk assessment modelling in distributed environments," *Journal of Systems and Software*, vol. 88, pp. 42–53, 2014. 13, 84

[39] G. Elliott, *Global Business Information Technology: An Integrated Systems Approach*. Pearson Education, 2004. 13

[40] D. L. Parnas and P. C. Clements, "A Rational Design Process: How and Why to Fake It," *IEEE transactions on software engineering*, no. 2, pp. 251–257, 1986. 14

[41] S. McConnell, *Code Complete*. Pearson Education, 2004. 14

[42] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for Agile Software Development," 2001. 14

[43] Y. Sugimori, K. Kusunoki, F. Cho, and S. Uchikawa, "Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system," *The International Journal of Production Research*, vol. 15, no. 6, pp. 553–564, 1977. 14

[44] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall Upper Saddle River, 2002, vol. 1. 14

[45] H. Kniberg, "Kanban vs Scrum," *Crisp AB. Viitattu*, vol. 1, pp. 1–41, 2009. 14

[46] T. Lindberg, C. Meinel, and R. Wagner, "Design Thinking: A Fruitful Concept for IT Development?" in *Design thinking*. Springer, 2011, pp. 3–18. 14

[47] (2014) Best examples of companies using continuous deployment. [Online]. Available: http://bit.ly/1OZQ5SP 15

[48] (2017) Crowdsourcing: A Defintion. [Online]. Available: http://bit.ly/QwOkEh 15

[49] M. Nebeling, M. Speicher, M. Grossniklaus, and M. C. Norrie, "Crowdsourced Web Site Evaluation with Crowdstudy," in *International Conference on Web Engineering*. Springer, 2012, pp. 494–497. 15

[50] M. Vukovic, "Crowdsourcing for Enterprises," in *Services-I, 2009 World Conference on*. IEEE, 2009, pp. 686–692. 15

[51] D. Liu, R. G. Bias, M. Lease, and R. Kuipers, "Crowdsourcing for Usability Testing," *Proceedings of the American Society for Information Science and Technology*, vol. 49, no. 1, pp. 1–10, 2012. 16

[52] S. Zogaj, U. Bretschneider, and J. M. Leimeister, "Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary," *Journal of Business Economics*, vol. 84, no. 3, pp. 375–405, 2014. 16

[53] (2015) Bug bounty program. [Online]. Available: http://bit.ly/2DF6k8z 16

[54] (2018) The Bug Bounty List. [Online]. Available: https://bugcrowd.com/list-of-bug-bounty-programs 16

[55] D. E. Knuth. (2015) Homepage. [Online]. Available: http://stanford.io/1RvP5se 16

[56] D. J. Bernstein. (2015) Homepage. [Online]. Available: http://bit.ly/2IzEnSK 16

[57] W. Harrison, "Eating Your Own Dog Food," *IEEE Software*, vol. 23, no. 3, pp. 5–7, 2006. 17

[58] A. Moskowitz, "Eat your own dog food," *; login:: the magazine of USENIX & SAGE*, vol. 28, no. 5, pp. 18–19, 2003. 17

[59] E. Schmidt and H. Varian. (2005) Google: Ten Golden Rules. [Online]. Available: http://bit.ly/2uoHSUl 17

[60] A. Prlić and J. B. Procter, "Ten Simple Rules for the Open Development of Scientific Software," *PLoS Comput Biol*, vol. 8, no. 12, p. e1002802, 2012. 17

[61] N. Jackson, J. Winn *et al.*, "Eating Your Own Dog Food," 2012. 17

[62] (2017) Continuous integration. [Online]. Available: http://bit.ly/1Ty3ZfV 17

[63] P. Brooks, B. Robinson, and A. M. Memon, "An Initial Characterization of Industrial Graphical User Interface Systems ," in *Software Testing Verification and Validation, 2009. ICST'09. International Conference on.* IEEE, 2009, pp. 11–20. 18

[64] E. Moritz, "Case study: How analysis of customer found defects can be used by system test to improve quality," in *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on.* IEEE, 2009, pp. 123–129. 18

[65] M. Gittens, H. Lutfiyya, M. Bauer, D. Godwin, Y. W. Kim, and P. Gupta, "An Empirical Evaluation of System and Regression Testing," in *Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research.* IBM Press, 2002, p. 3. 18

[66] J. D. Musa, "Software reliability-engineered testing," *Computer*, vol. 29, no. 11, pp. 61–68, 1996. 18

[67] M. Sullivan and R. Chillarege, "A comparison of software defects in database management systems and operating systems," in *Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers., Twenty-Second International Symposium on.* IEEE, 1992, pp. 475–484. 19

[68] E. N. Adams, "Optimizing Preventive Service of Software Products," *IBM Journal of Research and Development*, vol. 28, no. 1, pp. 2–14, 1984. 19

[69] L. M. Riungu, O. Taipale, and K. Smolander, "Research Issues for Software Testing in the Cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on.* IEEE, 2010, pp. 557–564. 19

[70] P. Chebyshev, "On Two Theorems Concerning Probability," *Zap. Akad. Nauk*, 1887. 20

[71] R. A. Fisher, "Theory of Statistical Estimation," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 22, no. 5. Cambridge University Press, 1925, pp. 700–725. 20, 140, 141, 142

[72] S. S. Wilks, "The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses," *The Annals of Mathematical Statistics*, vol. 9, no. 1, pp. 60–62, 1938. 20, 140, 141, 142

[73] H. Cramér, *On the composition of elementary errors.* Almqvist & Wiksells, 1928. 20

[74] R. Von Mises, "Statistik und Wahrheit," *Julius Springer*, 1928. 20

[75] N. Smirnov, "Table for Estimating the Goodness of Fit of Empirical Distributions," *The annals of mathematical statistics*, vol. 19, no. 2, pp. 279–281, 1948. 20

[76] T. W. Anderson and D. A. Darling, "A Test of Goodness of Fit," *Journal of the American statistical association*, vol. 49, no. 268, pp. 765–769, 1954. 20

[77] ——, "Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes," *The annals of mathematical statistics*, pp. 193–212, 1952. 20, 90, 103, 116

[78] B. M. Hill *et al.*, "A Simple General Approach to Inference About the Tail of a Distribution," *The annals of statistics*, vol. 3, no. 5, pp. 1163–1174, 1975. 21, 116

[79] J. Pickands III, "Statistical Inference Using Extreme Order Statistics," *The Annals of Statistics*, pp. 119–131, 1975. 21, 116

[80] J. Nair, A. Wierman, and B. Zwart, "The fundamentals of heavy-tails: properties, emergence, and identification," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1. ACM, 2013, pp. 387–388. 21, 116

[81] J. Mullahy, "Specification and testing of some modified count data models," *Journal of econometrics*, vol. 33, no. 3, pp. 341–365, 1986. 21, 116, 140, 142

[82] M. Rosenblatt *et al.*, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956. 22, 116, 140, 141

[83] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962. 22, 116, 140, 141

[84] V. A. Epanechnikov, "Non-Parametric Estimation of a Multivariate Probability Density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, 1969. 22, 116

[85] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. CRC press, 1986, vol. 26. 22, 116, 141

[86] S. J. Sheather and M. C. Jones, "A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 683–690, 1991. 22, 116, 140

[87] F. Galton, "Kinship and Correlation," *The North American Review*, vol. 150, no. 401, pp. 419–431, 1890. 22, 92, 103

[88] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of Internet chat systems," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003, pp. 51–64. 23, 114, 149

[89] M. Lukasik, P. Srijith, T. Cohn, and K. Bontcheva, "Modeling Tweet Arrival Times using Log-Gaussian Cox Processes." in *EMNLP*, 2015, pp. 250–255. 24, 114, 149

[90] C. Vande Kerckhove, B. Gerencsér, J. M. Hendrickx, and V. D. Blondel, "Markov modeling of online inter-arrival times," *arXiv preprint arXiv:1509.04857*, 2015. 24, 114, 149

[91] N. M. Markovitch and U. R. Krieger, "Nonparametric estimation of long-tailed density functions and its application to the analysis of World Wide Web traffic," *Performance Evaluation*, vol. 42, no. 2, pp. 205–222, 2000. 24, 114, 149

[92] R. E. Maiboroda and N. M. Markovich, "Estimation of heavy-tailed probability density function with application to Web data," *Computational Statistics*, vol. 19, no. 4, p. 569, 2004. 24, 114, 149

[93] (2017) Visualizing Inter-Arrival Times of Tweets. [Online]. Available: http://bit.ly/2tm8AMe 24, 114, 149

[94] P. Burnap, M. L. Williams, L. Sloan, O. Rana, W. Housley, A. Edwards, V. Knight, R. Procter, and A. Voss, "Tweeting the terror: modelling the social media reaction to the Woolwich terrorist attack," *Social Network Analysis and Mining*, vol. 4, no. 1, p. 206, 2014. 25, 114, 149

[95] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959. 25

[96] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013. 25

[97] F. Provost and R. Kohavi, "Glossary of terms," *Journal of Machine Learning*, vol. 30, no. 2-3, pp. 271–274, 1998. 25

[98] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT press, 2012. 25

[99] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in *The elements of statistical learning*. Springer, 2009, pp. 485–585. 25

[100] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1. 26

[101] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, NY, USA:, 2001, vol. 1, no. 10. 26, 27

[102] S. Piramuthu and R. T. Sikora, "Iterative feature construction for improving inductive learning algorithms," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3401–3406, 2009. 26

[103] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial Intelligence: A Modern Approach*. Prentice hall Upper Saddle River, 2003, vol. 2, no. 9. 26

[104] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian Approach to Filtering Junk E-mail," in *Learning for Text Categorization: Papers from the 1998 workshop*, vol. 62, 1998, pp. 98–105. 26

[105] J. R. Quinlan, "Induction of Decision Trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986. 26

[106] Y. Bengio, O. Delalleau, and C. Simard, "Decision trees do not generalize to new variations," *Computational Intelligence*, vol. 26, no. 4, pp. 449–467, 2010. 26

[107] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995. 27

[108] M. A. Aizerman, "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning," *Automation and remote control*, vol. 25, pp. 821–837, 1964. 27

[109] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152. 27

[110] E. Osuna, R. Freund, and F. Girosit, "Training Support Vector Machines: an Application to Face Detection," in *Computer vision and pattern recognition, 1997. Proceedings., 1997 IEEE computer society conference on*. IEEE, 1997, pp. 130–136. 27

[111] L. Liao and W. S. Noble, "Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships," *Journal of computational biology*, vol. 10, no. 6, pp. 857–868, 2003. 27

[112] T. K. Ho, "Random decision forests," in *Document analysis and recognition, 1995., proceedings of the third international conference on*, vol. 1. IEEE, 1995, pp. 278–282. 27

[113] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996. 27

[114] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT press, 1999. 28

[115] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Pearson London, 2014, vol. 3. 28

[116] J. J. Webster and C. Kit, "Tokenization as the initial phase in NLP," in *Proceedings of the 14th conference on Computational linguistics-Volume 4*. Association for Computational Linguistics, 1992, pp. 1106–1110. 28

[117] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge university press, 2014. 28

[118] (2017) Stopword Lists. [Online]. Available: http://bit.ly/2jwKvDa 28

[119] H. P. Luhn, "Key word-in-context index for technical literature (kwic index)," *Journal of the Association for Information Science and Technology*, vol. 11, no. 4, pp. 288–295, 1960. 28

[120] J. B. Lovins, "Development of a Stemming Algorithm," *Mech. Translat. & Comp. Linguistics*, vol. 11, no. 1-2, pp. 22–31, 1968. 28

[121] D. Manning, "Introduction," in *Introduction to Industrial Minerals*. Springer, 1995, pp. 1–16. 28

[122] H. Kučera and W.-N. Francis, *Computational analysis of present-day American English*. Dartmouth Publishing Group, 1967. 29

[123] T. K. Landauer, P. W. Foltz, and D. Laham, "An Introduction to Latent Semantic Analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998. 29

[124] T. Hofmann, "Probabilistic Latent Semantic Analysis," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 289–296. 29

[125] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003. 29

[126] A. G. Jivani *et al.*, "A Comparative Study of Stemming Algorithms," *Int. J. Comp. Tech. Appl*, vol. 2, no. 6, pp. 1930–1938, 2011. 30

[127] N. Naveed, T. Gottron, J. Kunegis, and A. C. Alhadi, "Searching Microblogs: Coping with Sparsity and Document Quality," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 183–188. 30

[128] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A Biterm Topic Model for Short Texts," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 1445–1456. 30

[129] J. Yin and J. Wang, "A Dirichlet multinomial mixture model-based approach for short text clustering," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 233–242. 30

[130] V. K. R. Sridhar, "Unsupervised Topic Modeling for Short Texts Using Distributed Representations of Words," in *VS@ HLT-NAACL*, 2015, pp. 192–200. 30

[131] Y. Zuo, J. Wu, H. Zhang, H. Lin, F. Wang, K. Xu, and H. Xiong, "Topic Modeling of Short Texts: A Pseudo-Document View," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 2105–2114. 31

[132] A. Schofield and D. Mimno, "Comparing Apples to Apple: The Effects of Stemmers on Topic Models," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 287–300, 2016. 31

[133] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002. 31

[134] J. Morris and G. Hirst, "Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text," *Computational linguistics*, vol. 17, no. 1, pp. 21–48, 1991. 32

[135] H. Kozima, "Text Segmentation Based on Similarity Between Words," in *Proceedings of the 31st annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1993, pp. 286–288. 32

[136] J. C. Reynar, "An Automatic Method of Finding Topic Boundaries," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994, pp. 331–333. 32

[137] D. Beeferman, A. Berger, and J. Lafferty, "Statistical Models for Text Segmentation," *Machine learning*, vol. 34, no. 1-3, pp. 177–210, 1999. 32

[138] M. Galley, K. R. McKeown, E. Fosler-Lussier, and H. Jing, "Discourse segmentation of multi-party conversation," in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003. 32

[139] V.-A. Nguyen, J. Boyd-Graber, and P. Resnik, "SITS: A Hierarchical Nonparametric Model using Speaker Identity for Topic Segmentation in Multiparty Conversations," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 78–87. 32

[140] M. Brooks, K. Kuksenok, M. K. Torkildson, D. Perry, J. J. Robinson, T. J. Scott, O. Anicello, A. Zukowski, P. Harris, and C. R. Aragon, "Statistical Affect Detection in Collaborative Chat," in *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 317–328. 32

[141] A. P. Schmidt and T. K. Stone, "Detection of Topic Change in IRC Chat Logs," 2013. 32

[142] D. C. Uthus and D. W. Aha, "Multiparticipant chat analysis: A survey," *Artificial Intelligence*, vol. 199, pp. 106–121, 2013. 32

[143] P. Muller, S. Devnani, J. Julius, D. Gagliardi, C. Marzocchi, R. Ramlogan, and D. Cox. (2016) Annual report on European SMEs 2015/2016. [Online]. Available: http://bit.ly/2vNmqfk 36, 37

[144] (2014) Annual Report on European SMEs 2014/2015. [Online]. Available: http://bit.ly/2FSPGn3 37

[145] H. H. Olsson, H. Alahyari, and J. Bosch, "Climbing the 'Stairway to Heaven'–A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software," in *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*. IEEE, 2012, pp. 392–399. 37

[146] O. T. Pusatli and S. Misra, "A discussion on assuring software quality in small and medium software enterprises: An empirical investigation," *Tehnički vjesnik*, vol. 18, no. 3, pp. 447–452, 2011. 37

[147] (2016) Eating your own dog food. [Online]. Available: http://bit.ly/2kHVTHP 38

[148] (2014) Not Eating Your Own Dog Food? You Probably Should Be. [Online]. Available: http://bit.ly/2k64eWH 38

[149] (2013) Clinging to Outlook. [Online]. Available: http://bit.ly/19dZ40P 38

[150] M. Sullivan and R. Chillarege, "Software Defects and their Impact on System Availability: A Study of Field Failures in Operating Systems," in *FTCS*, vol. 21, 1991, pp. 2–9. 48

[151] B. Boehm and V. R. Basili, "Software Defect Reduction top 10 list," *Foundations of empirical software engineering: the legacy of Victor R. Basili*, vol. 426, p. 37, 2005. 48

[152] G.-A. Klutke, P. C. Kiessler, and M. A. Wortman, "A critical look at the bathtub curve," *IEEE Transactions on Reliability*, vol. 52, no. 1, pp. 125–129, 2003. 50

[153] (2018) The Bathtub Curve and Product Failure Behavior. [Online]. Available: http://bit.ly/2Ix2em5 50

[154] J. Dunne, D. Malone, and J. Flood, "Social testing: A framework to support adoption of continuous delivery by small medium enterprises," in *2015 Second International Conference on Computer Science, Computer Engineering, and Social Media (CSCESM)*, Sept 2015, pp. 49–54. 55

[155] (2017) About Early Technology Adoption (Dogfooding). [Online]. Available: https://bit.ly/2ycn4Gn 57

[156] T. Maillart, M. Zhao, J. Grossklags, and J. Chuang, "Given enough eyeballs, all bugs are shallow? revisiting eric raymond with bug bounty programs," *Journal of Cybersecurity*, vol. 3, no. 2, pp. 81–90, 2017. 58

[157] P. Muller, C. Caliandro, V. Peycheva, D. Gagliardi, C. Marzocchi, R. Ramlogan, and D. Cox. (2015) SME performance review European SME's. [Online]. Available: http://bit.ly/23NnKIx 61, 86

[158] K. Pearson, "Contributions to the Mathematical Theory of Evolution," *Philosophical Transactions of the Royal Society of London. A*, vol. 185, pp. 71–110, 1894. 64

[159] M. L. Delignette-Muller and C. Dutang, "fitdistrplus: An R package for fitting distributions," *Journal of Statistical Software*, vol. 64, no. 4, pp. 1–34, 2015. [Online]. Available: http://www.jstatsoft.org/v64/i04/ 64, 90, 116

[160] C. J. G. Bellosta. Package ADGofTest. [Online]. Available: http://bit.ly/1NU3c5y 64, 90, 116

[161] E. A. Elsayed, *Reliability Engineering*. John Wiley & Sons, 2012, vol. 88. 80

[162] (2017) New Relic - Application and Performance Monitoring. [Online]. Available: http://bit.ly/2gpkvYm 86

[163] (2017) IBM Operations Analytics - Predictive Insights. [Online]. Available: https://ibm.co/2G1EdoT 86

[164] (2017) Ruxit - Cloud native monitoring. [Online]. Available: http://bit.ly/2mKwAFJ 86

[165] D. Blackwell *et al.*, "A renewal theorem," *Duke math. J*, vol. 15, pp. 145–150, 1948. 87

[166] W. S. Jewell, "A Simple Proof of: L= λ W," *Operations Research*, vol. 15, no. 6, pp. 1109–1116, 1967. 87

[167] (2011) Calculating the Cost of Data Center Outages. [Online]. Available: http://bit.ly/2ppGMGW 87

[168] Y. Alshamaila, S. Papagiannidis, and F. Li, "Cloud computing adoption by SMEs in the north east of England: A multi-perspective framework," *Journal of Enterprise Information Management*, vol. 26, no. 3, pp. 250–275, 2013. 88

[169] T. Oliveira, M. Thomas, and M. Espadanal, "Assessing the determinants of Cloud computing adoption: An analysis of the manufacturing and services sectors," *Information & Management*, vol. 51, no. 5, pp. 497–510, 2014. 88

[170] M. F. Gholami, F. Daneshgar, G. Low, and G. Beydoun, "Cloud migration process—A survey, evaluation framework, and open challenges," *Journal of Systems and Software*, vol. 120, pp. 31–69, 2016. 88

[171] (2015) Executive summary - Final report - Annual Report on European SMEs - 2014 / 2015 - SMEs start hiring again. [Online]. Available: http://bit.ly/2GHYKg3 90

[172] R. A. Fisher, "Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population," *Biometrika*, vol. 10, no. 4, pp. 507–521, 1915. 92, 103

[173] C. Spearman, "The Proof and Measurement of Association between Two Things," *The American journal of psychology*, vol. 15, no. 1, pp. 72–101, 1904. 92, 103

[174] G. E. Box and D. A. Pierce, "Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Mod-

els," *Journal of the American statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970. 92, 103

[175] Test for Association/Correlation Between Paired Samples. [Online]. Available: http://bit.ly/2djPSA7 92

[176] Fitting Linear Models. [Online]. Available: http://bit.ly/2dvqYet 92, 157

[177] Auto and Cross-Covariance and Correlation Function Estimation. [Online]. Available: http://bit.ly/2dKfLZl 92

[178] R. A. Fisher, "On the interpretation of χ 2 from Contingency Tables, and the Calculation of P," *Journal of the Royal Statistical Society*, vol. 85, no. 1, pp. 87–94, 1922. 93, 103

[179] ——, *Statistical Methods for Research Workers*. Genesis Publishing Pvt Ltd, 1925. 93, 103

[180] Fisher's Exact Test for Count Data. R Package. [Online]. Available: http://bit.ly/1NU3c5y 93

[181] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*. Springer, 2011. 94

[182] M. B. Wilk and R. Gnanadesikan, "Probability plotting methods for the analysis for the analysis of data," *Biometrika*, vol. 55, no. 1, pp. 1–17, 1968. 94

[183] B. C. Arnold, *Pareto Distribution*. Wiley Online Library, 2015. 104

[184] Y.-S. Chen, P. Pete Chong, and Y. Tong, "Theoretical foundation of the 80/20 rule," *Scientometrics*, vol. 28, no. 2, pp. 183–204, 1993. 104

[185] G. Apostolakis, S. Garribba, and G. Volta, *Synthesis and Analysis Methods for Safety and Reliability Studies*. Springer, 1980, vol. 106. 105

[186] M. M. Ananda, "Confidence intervals for steady state availability of a system with exponential operating time and lognormal repair time," *Applied Mathematics and Computation*, vol. 137, no. 2, pp. 499–509, 2003. 105

[187] M. M. Ananda and J. Gamage, "On steady state availability of a system with lognormal repair time," *Applied mathematics and computation*, vol. 150, no. 2, pp. 409–416, 2004. 105

[188] M. Pendolino. (2017) 3 Ways Collaborative Software Can Solve Enterprise Challenges. [Online]. Available: http://bit.ly/2uzU480 112

[189] K. Wolf. (2017) 8 Business Problems the Best Collaboration Software Can Solve. [Online]. Available: http://bit.ly/2vfreHX 112

[190] M. Haughey. (2017) Setting up Slack for small teams. [Online]. Available: http://bit.ly/2vzw81Q 112

[191] "Watson Workspace," 2017. [Online]. Available: https://ibm.co/2uG4ZgW 113

[192] "Slack," 2017. [Online]. Available: http://bit.ly/1uEVWVc 113

[193] "Microsoft Teams," 2017. [Online]. Available: http://bit.ly/2ffnmz5 113

[194] "Azendoo," 2017. [Online]. Available: http://bit.ly/1lnHcX5 113

[195] C. A. Sottile, "Sick of Email? Slack Wants to Kill Your Inbox Clutter - NBC News," 2017. [Online]. Available: http://nbcnews.to/2uyzLYR 113

[196] C. Fowler, "How To Avoid Email Paralysis," 2017. [Online]. Available: http://bit.ly/2vhWlT1 113

[197] "How the engineering team at IBM uses Slack throughout the development lifecycle," 2017. [Online]. Available: http://bit.ly/2qcd07G 113

[198] C. Boulton, "How DevOps, agile spurred Slack enterprise adoption," 2017. [Online]. Available: http://bit.ly/2kTMHB8 113

[199] (2017) Ubuntu IRC Logs. [Online]. Available: https://irclogs.ubuntu.com/ 115, 154, 173

[200] M. Elsner and E. Charniak, "Disentangling chat," *Computational Linguistics*, vol. 36, no. 3, pp. 389–409, 2010. 115

[201] R Package Density. [Online]. Available: http://bit.ly/2DFsSoX 117

[202] M. et al. R Package vcd. [Online]. Available: http://bit.ly/2vyTULd 118

[203] A. Z. Christian Kleiber. R Package AER. [Online]. Available: http://bit.ly/1LzY9pI 119

[204] W. A. Huber. What to do when count data does not fit a Poisson distribution. [Online]. Available: http://bit.ly/2uHCkIx 150

[205] (2015) We just don't speak anymore. [Online]. Available: http://bit.ly/2yDXzJ6 152

[206] (2015) 73 Texting Statistics. [Online]. Available: http://bit.ly/2kjHeF8 152

[207] (2017) The Six Benefits of Real-Time Chat For Your Mobile Workforce. [Online]. Available: http://bit.ly/2GZ7Pk9 152

[208] (2018) The Advantages of a Chat Room. [Online]. Available: http://bit.ly/2iy5qVS 152

[209] (2017) Can We Chat? Instant Messaging Apps Invade the Workplace. [Online]. Available: http://bit.ly/2Egv3ES 152

[210] (2016) How to Deal With Social Media Overwhelm. [Online]. Available: http://bit.ly/2yN5e8r 152

[211] (2016) Expect more chatbots. [Online]. Available: http://bit.ly/2z771cJ 152

[212] (2017) Social Messaging: Catalysing The Next Wave of Digital Revolution In Communication. [Online]. Available: http://bit.ly/2FekIpz 152

[213] (2017) Pros and Cons of corporate group chats. [Online]. Available: http://bit.ly/2Eda9m0 152

[214] (2017) Is group chat making you sweat? [Online]. Available: http://bit.ly/1pupgj8 152

[215] (2017) The Value and Benefits of Text Mining. [Online]. Available: http://bit.ly/2zJcDcl 152

[216] (2017) Gain business insight with Big Data. [Online]. Available: http://bit.ly/2zPxmcC 152

[217] (2015) Improving the Consumer E-commerce Experience Through Text Mining. [Online]. Available: http://bit.ly/2z8eYyv 152

[218] L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text Line Segmentation of Historical Documents: a Survey," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 123–138, 2007. 153

[219] J. Weisz, "Segmentation and Classification of Online Chats," 2008. 153

[220] (2017) Group-chat software sees explosive growth and intense competition. [Online]. Available: http://bit.ly/2nOuaJ8 153

[221] (2017) Qualitative Sample Size. [Online]. Available: http://bit.ly/2hWeh3R 156

[222] E. Dale and J. S. Chall, "A Formula for Predicting Readability: Instructions," *Educational research bulletin*, pp. 37–54, 1948. 157

[223] M. Coleman and T. L. Liau, "A computer readability formula designed for machine scoring." *Journal of Applied Psychology*, vol. 60, no. 2, p. 283, 1975. 157

[224] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, "Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel," Naval Technical Training Command Millington TN Research Branch, Tech. Rep., 1975. 157

[225] R. Gunning, "The Technique of Clear Writing," 1952. 157

[226] Fitting Generalized Linear Models. [Online]. Available: https://bit.ly/2MDtmRM 157

[227] D. Biber, S. Conrad, and R. Reppen, *Corpus Linguistics: Investigating Language Structure and Use*. Cambridge University Press, 1998. 173

[228] (2017) Lancsbox: Lancaster University corpus toolbox. [Online]. Available: http://bit.ly/2smurrz 173

[229] K. Huotari and J. Hamari, "Defining Gamification - A Service Marketing Perspective," in *Proceeding of the 16th International Academic MindTrek Conference*. ACM, 2012, pp. 17–22. 185

[230] F. I. John, "Single Server Queues with Dependent Service and Inter-Arrival Times," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 3, pp. 526–534, 1963. 186

[231] (2017) World Trade Report. [Online]. Available: http://bit.ly/2dvgC2y 188