UNIVERSITY OF DUBLIN, TRINITY COLLEGE

The Search for the Searcher: User Identification using Web Search Log Mining

Author:
Tom Moore

Supervisor:
Professor Séamus Lawless

A thesis submitted in fulfilment of the requirements for the degree of MSc in Computer Science

to the

University of Dublin, Trinity College

May 18, 2017

Declaration of Authorship

I, Tom Moore, declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work. I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed: Tom Moore

Date: 18th of May 2017

Abstract

User grouping is the act of treating users that share common characteristics as groups that can be modeled. These characteristics can be called user features. Transactional logs are records of interactions between users and systems. User features can be derived from transactional logs. It must therefore be possible to derive user groups from transactional logs. One method of forming groups from data points is with data clustering. As user features are shared characteristics about users, it may be possible to form groups of users using data clustering techniques on their use features.

However, is it possible to extract user features from transactional logs and then cluster those features using data clustering techniques to form clusters from which user groups can be derived? This thesis is an experiment to see if such an idea is possible.

To that end this dissertation proposes a pipeline that will take transactional logs as an input and will output data clusters from which user groups can be derived. The outputted clusters will be validated and analysed using data analysis techniques such as silhouette scores to see if it is possible to derive suer groups from them.

Acknowledgements

I would like to first thank my supervisor Séamus Lawless for his help, guidance and enthusiasm throughout the course of this project.

I would also like to extend that same thanks to Gary Munnelly and Annalina Caputo. All three of them helped greatly throughout this project. Their unfaltering support has proved invaluable.

Finally, I would also like to acknowledge the support of my family and friends throughout the year. In particular my parents, Patricia and Richard and my brother Sam.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The interaction between users(humans) and systems(computers is called Human Computer Interaction (HCI). In HCI, user models are the models systems have of users that reside inside of their environments (Fischer, 2001). These models form representations of the users that system administrators and architects can use to alter the system.

User modelling is a broad topic with many sub categories of models. It is thus necessary to know what is the best model to use for a system in question. This is dependant entirely on the amount of information that one wishes to know about that systems' users. It is also dependant on the type of information about the users that one wishes to know.

Due to this, the source of the information that is used to model the system is very important. An example would be to compare transactional log data mining and polling the users.

Polling allows system administrators to get direct feedback of their systems. While this is a fine method for a large business with a large amount of user interaction, such as a Facebook, it is less useful for smaller businesses or other systems. This is because polls are both expensive and will only reveal information that users either choose to reveal or are asked about.

However, by using transactional log data, it is possible to extract a user's interactions with the system in question. Transactional logs are the records of interactions (or transactions) between two systems. In the case of websites these interactions are between their systems and their users. These transactions can hold hidden information, such as behavioural trends amongst group of users. Such information would not be possible to gain from a poll.

However, for log data to become information, it must first be manipulated and mined. For it to become knowledge, that information must be interpreted. This interpenetration is part of this projects motivation.

By mining transactional log data it is possible to get characteristics of its users. To take this information and convert to knowledge requires an extra leap. To bridge the gap between transactional log data, and user models, or more particularly user groups is a key motivating factor for this thesis.

By understanding these models it is possible to learn more about the users beyond how they act with the system. It is possible to extract the knowledge a user has of what is contained within the systems domain.

Part of what motivates the user modelling for this project is the need to understand the users' knowledge of the domain they are accessing. What do they know about the data stored in the system? Or about specific topics? Where do their general interests lie regarding the type of data?

By modelling the users the hope is that these questions can be answered. These answers can then be used to further improve the system in question. If such a method proves reliable, it can be repeated with other datasets. If this reliability were to extend beyond its initial dataset, then a method of improving systems without extra user interaction may be created.

1.2 Research Question

Having stated the motivation that drives this thesis, it is now necessary to ask what is the full purpose of the project? To what end is the research and implementation of this project directed? Transactional logs are a common feature in the world of web based systems. For such systems user grouping is an effective way of gaining a greater insight into their users. A greater insight may lead to major system improvements for users.

With that knowledge in hand, the research question to be pursued:

“Can a Pipeline be Defined and Implemented that allows for the Creation of User Groups from Transactional Log Data?”

The reasoning behind this question can be found by breaking down the title of this thesis into questions and then answering them. The title of this thesis is"The Search for the Searcher: User Identification using Web Search Log Mining". The questions that are derived are:"What is web search log mining?","What is meant by 'user identification'?" and finally"Can a replicable method be found to achieve 'user identification' using web search log mining?"

All records of a users interactions with a web based system are held in transactional logs. These logs contain information regarding the queries used by a user as well as how they acted with the information that was returned from using those queries(). The information store that contains a users interaction with a web based system are those system's transactional logs.

It is these logs that are the web search logs in question. To extract data from these web search logs is to"mine" them. By"mining" them successfully it is possible to get information about the users. This information can then be used to"identify" them.

"Identifying" the users in the context of this project means to successfully model them. As already stated, there are multiple methods of modelling users to greater and lesser abstractions. The question remains as to which method of modelling the users is most effective. The greater the information extracted from a user the greater the level of"identification" that is possible.

By compiling the information from the disparate users together, it is possible to achieve a more specific model of the group. To that end by defining"User Groups" it may be possible to successfully model the users of this system. User groups are groups of users that share common characteristics that can be used to typify later users, who would then be added to such a group (Kay, 2001). By defining and populating these groups, one can model and"identify" the users successfully.

When speaking of a replicable method, one must first break any such method down into its constituent parts. The information being used for identification may not be in a form suited for that task. A method to parse that data into a usable form is therefore necessary. Operations must then be performed on the data in order to successfully extract the groups necessary to model the users. These operations performed in sequence would naturally form a pipeline. This pipeline would use transactional log data as an input and output clustered data from which the user groups can be formed. By combining smaller methods such as these together a pipeline begins to appear.

By answering those three questions a new one is formed. Which is the question posed at the commencement of this section namely - "Can a Pipeline be Defined and Implemented that allows for the Creation of User Groups from Transactional Log Data?".

1.3 Goals and Objectives

The goal of this project is to create a pipeline through which transactional log data can be converted into user groups. That is, to create a pipeline that will take transactional logs as an input and output data that can be used to create user groups. However, how this is achieved is discussed here. The objectives that will allow completion of this project are:

1. **To Store and Process the logs so as to make them Usable**

2. **To extract the Pertinent Data from the logs and Process it**

3. **To Define and Extract User Features from that Data**

4. **To Cluster the Users of the System based on those Features**

5. **To Validate and Analyse the Clusters that formed**

6. **To use that Analysis to derive User Groups if Possible**

1.4 Overview

So far the main concepts defining the foundations of this project have been discussed. These include the underlying motivations for the project, the research question it hopes to answer and the goals and objectives that are set to be achieved. The chapters that follow, will expand on the concepts and approaches as follows:

Chapter 2 discusses the state-of-the-art which provides a solid grounding in the current state, progress and performance of similar projects to date. This chapter analyses current modelling methods and explores the stereotyping of users based on their transactional log data.

Chapter 3 describes the design methodology behind the various stages of the project. This includes the definition of the pipeline, as well as its various parts.

Chapter 4 outlines the implementation of the design and details the technical aspects of the pipeline in greater depth. The topics discussed in this chapter include the setting up of the test environment and the implementation of the various aspects of the pipeline.

Chapter 5 evaluates the results of the project. This chapter critically analyses the experiments and tests, as well as the results achieved from the project in order to evaluate whether the pipeline is both feasible on other systems as well as useful.

Chapter 6 has some final concluding thoughts on the project as a whole. This chapter discusses the achievements of the project, its limitations and the obstacles which occurred in the pursuit of the goal of the project. Also discussed in this chapter is the future work to be explored and my recommendations for doing so.

Chapter 2

State of the Art

. At the beginning of this project a number of design choices regarding technologies and methodologies had to be made first in order to structure a feasible solution to the issues raised in the previous chapter. The discovery of these technologies and methodologies was accomplished through identifying applicable trends within the realms of user modelling, data clustering, and cluster analysis.

This chapter aims to discuss these various state-of-the-art approaches to problems similar to those found in this project. This will be done through the analysis of the technologies and methods applied, as well as the results that they produce.

2.1 User Modelling

One of the most important aspects of this project is the modelling of the users. User modelling, as a concept, dates back to the late 1970's when user information was stored and contained completely within an application. At that time little distinction was given between user modelling and other general application functions (Kobsa, 2001).

Today, there are multiple disparate forms of user modelling. When it comes to what models to use when modelling the users, there are a large amount of choices available.

One such method of user modelling is to stereotype the users. User stereotyping was proposed as a method of quickly building models of users using little data(Rich, 1979). Rich defined these stereotypes as a collection of attributes that often co-occur in people. Stereotypes solved the issue of the new user problem. This problem is the issue of defining individualised user models where there were no previous interactions from which to build.

Having a pool of stereotyped users can be extremely useful as they allow the modeller to make a large number of inferences from a small sample size. The drawback to user stereotypes however is that they require non-monotonic reasoning techniques (Rich, 1989).

Non-monotonic reasoning is a reasoning that creates an inference in which the reasoner draws their own tentative conclusions(Brewka, 1991). This allows the reasoner to withdraw their conclusions at a later date when newer more accurate information is available. As stereotypes require this reasoning it can be inferred that they are, by design, tentative models.

This means that such stereotypes are understood to be possibly flawed from the use of the term stereotype. This flawed nature is one reason why a decision was made not to attempt to create stereotypes. While they do not require much data, they do require a large amount of scepticism. If chosen user stereotypes were used for the modelling paradigm of the project, one could not be sure of the usefulness of the results.

An alternative user modelling approach to grouping users according to their common characteristics is to group them according to their common interests in a community (Paliouras et al., 1999). This is an explicit model of the shared interests of the users.

Due to the differences in their definitions, it can be assumed that stereotypes and communities are derived differently. This is accurate. Stereotypes are generally constructed by individuals (Paliouras et al., 1999) as opposed to by a computer. This means, however, that while they hold the advantages laid out above, they are inherently influenced by their designers own bias. This leads to the aforementioned issue with the non-monotonic reasoning that is used in stereotype construction, namely the stereotypes may be inaccurate or simply false.

Conversely, communities are mostly created using unsupervised learning and clustering (Paliouras et al., 1999). This means that communities tend to be more driven by the dataset as opposed to the bias of the modeller. This can lead to a more accurate reflection of the common features of the users.

Unlike user stereotyping, communities fall victim to the new user problem. This is because, unlike a stereotype, entry for a user into a community requires similarity in features to the already present community members. This similarity cannot be obtained without feedback from the user as to their features. Features in this case might also be called interests, or whatever term that describes the common aspects of the users.

Stereotyping is unsuitable as a model due to its inability to be accurate and unbiased. Communities are also unsuitable due to their inability to solve the new user issue. The new user issue is imperative to solve as there will be no initial users to form communities with whose features (or interests) are of a known quantity. For these reasons, user grouping was chosen, as the modelling style for this project.

User grouping, or group modelling, is the modelling of a group of users in order to serve them as a group, rather than as individuals. This can be extremely useful for settings in which users might interact with systems of particular types simultaneously. An example of this would be a media system such as Youtube, where multiple users may watch the same video at the same time. Having a user group of those users would allow Youtube to recommend videos enjoyed by other members of the same group.

User stereotyping is considered to be a form of user grouping. This is because it captures common features of groups of users. However, there is a tendency for them to be used in different ways. The main difference being that stereotypes have users belonging to multiple groups, whereas a user group have no users in multiple groups.

Group models are meant to serve as models of groups of users that use systems. It can be understood then, that the aim of group modelling is not to solve the new user problem. However there are suggestions that it may in fact be able to do so (Masthoff, 2003). This is because any new user joining a group will have similar features to those already present.

As group modelling hypothetically solves the new user issue it stands to reason that for this project it would be most ideal. This is due to the idea of deriving the groups or any other model from the data post interaction, as opposed to in real time. So all users will at some point be new users. It is also because the models that are to be derived must be exclusive. This is to ensure that the users in question are not double counted when it comes to altering the system to their benefit. This double counting may lead to inaccuracies in the ability of the new system to deliver what the users need.

2.1.1 User Features

To derive any form of user models it is necessary to have relevant data from the users in question. To that end, as the data in use in this project are user interactions with a web based archive, the knowledge the user has of the archive is what can be considered to be of most interest.

In 2010, Kamps and Zhang published a paper that used "User features" to stereotype their userbase (Kamps and Zhang, 2010). Their dataset was transactional logs that detailed user interactions with a web based historical archive which bears a broad similarity to my own dataset.

Kamps and Zhang, defined two basic stereotypes of their users. The first of these was the "novices" or users with low knowledge of the archive in question. The second was the "experts" or users with high knowledge of the archive. Having defined two stereotypes they proceeded to divide their users into these stereotypes on the basis of their relevant user features. These were different metrics of the users interactions with the archive. These “user features” were extracted from the transactional logs.

Due to the similar nature of the datasets it is not improper to assume that the users in the dataset of this project hold similar features. To that end, using some of the features used by Kamps and Zhang to stereotype their users may be the most appropriate course of action with regards to metrics around which to group the users for this project.

2.2 Data Clustering

In order to create the user groups in this project, it is necessary to automatically group the users according to the metrics by which the project will model them. It is from these groups, or clusters, that the models will be formed. To that end, clustering algorithms were explored as a basis for doing so.

Most of the examinations led to the view that the best choice for an algorithm was the K-means clustering algorithm. This is due in part to its versatility with large scale databases as well as the ready number of machine learning tools available to implement it efficiently (Pedregosa et al., 2011).

K-Means clustering aims to partition N data points into K clusters. In these clusters each data point belongs a single cluster. The cluster with the nearest mean serves as a prototype of the cluster. Each cluster is centred around a centroid M As a result the data space is partitioned into individual voronoi cells (Hartigan, 1975).

The algorithm for K-means is as follows:

1. Begin with initial guesses for cluster centres (centroids).

2. For each data point, find the cluster centre with the closest mean. This partitions the data points.

$$S_i^{(t)} = \{x_j : \|x_j - m_i^{(t)}\| \leq \|x_j - m_{i^*}^{(t)}\| \text{for all } i^* = 1..., k\} \quad (2.1)$$

3. Replace each centroid by the average of the data points in its partition.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x^j \varepsilon S_i^{(t)}} x_j \quad (2.2)$$

4. Finally, iterate the second and third steps until convergence.

Though K-means is a powerful tool, there are different additions to it that can strengthen its ability to cluster data. One of these is in the selection of initial values. Whilst traditional K-means uses random initial values there are algorithms that exist that aim to improve on this. One of these is K-means++.

A simplified K-means++ algorithm works as follows (Arthur and Vassilvitskii, 2009):

1. First choose one uniform centre randomly from among the data points.

2. Then, for each data point X, $D(x)$ is computed. This is the distance between X and the closest previously chosen centre.
3. A new random data point is then chosen as the new centre. This is done using a weighted probability distribution where the probability of new point x being chosen is proportional to $D(x)^2$.
4. Finally repeat the first and second step until k centres have been chosen.

This method of choosing the centres around which clusters are formed should decrease the possibility of sub optimal clustering.

In figure 2.1 the advantages of K-means++ over K-means can be

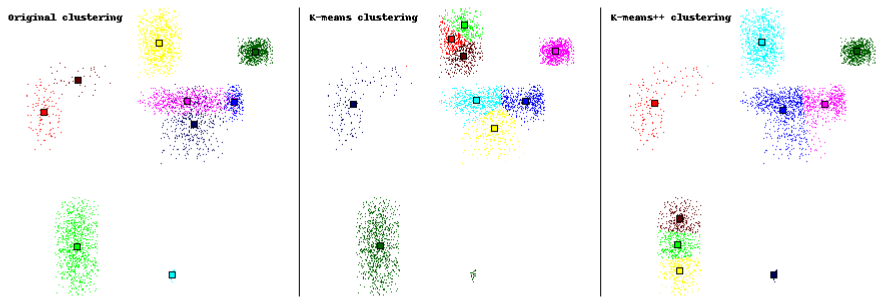

FIGURE 2.1: An example of unclustered data vs. K-Means and K-means++

(Ahlouche, 2013)

seen. K- Means++ shows the small light-blue cluster (in the first clustering) as a complete cluster. On the other hand, K-means sees it as part of a bigger cluster. Such inaccuracies could lead to errors in the grouping of users in this project.

2.3 Cluster Analysis

Having created the clusters it is necessary to analyse them so as to validate their integrity and derive the user groups. By analysing the resulting clusters, the hope is to be able to form groups of users that have relevance regarding their knowledge of the database.

The goal of cluster analysis is to identify groups that are formed in the data clustering process(Norusis, 2012). There is no distinct analysis algorithm for all cluster types. As there are many different clustering algorithms there can be no certain method of analysing clusters that are formed using different processes. That said, the majority of cluster evaluations can be split into separate categories. Two of these are external and internal evaluations Halkidi, Batistakis, and Vazirgiannis, 2001. External evaluations rely on unused external data to evaluate the clusters. Internal, on the other hand, relies only on the clustered data itself to form the evaluations Halkidi, Batistakis, and Vazirgiannis, 2001. In this project there is no unclustered data that can be used to externally evaluate the clustered data. As such, all the evaluations in this project are by nature internal.

While there are many forms of graphical and non-graphical internal cluster analysis methods in use, silhouette scores have been selected for the analysis and validation process in this project. This is because silhouette scores are both natively supported in the chosen machine learning library (Pedregosa et al., 2011) and they are also a clear metric of cluster validity that can be reliably graphed and interpreted.

"Silhouettes" were first introduced in 1987 as a graphical aid for interpretation and validation of cluster analysis (Rousseeuw, 1987). A silhouette score or silhouette coefficient, can be defined as a representation of cohesion versus separation in a numerical value. Cohesion is the measure of how closely related points are within a cluster. Separation is the measure of how distinct or separated a point within a cluster is from other clusters. As Silhouette coefficients are a measure of cohesion versus separation it provides a measure of how well a data point was classified when it was assigned to a cluster(Norusis, 2012). This can be determined graphically by both the tightness of the clusters and the separation between them.

Silhouette scores are calculated using the following formula shown in 2.3. Where $a(i)$ is the lowest average cohesiveness of the point i in a cluster and $b(i)$ is the lowest average separation of that point to any cluster of which i is not a member. Together, they calculate, $s(i)$, which is the silhouette score of that point in a cluster. This formula proves that $-1 > s(i) < 1$ for all points in a cluster. However, for a K-means cluster the Silhouette score cannot go below 0. This is because having a score of less than 0 means it has been assigned to the wrong cluster.

$$S(i) = \frac{b(i) - a(i)}{\max a(i), b(i)} \tag{2.3}$$

By using Silhouette scores, this project aims to both validate the clusters formed from the user features and to analyse those clusters to correctly determine the user groups.

Chapter 3

Design

3.1 Introduction

. The main goal of this project is to create a pipeline that will generate “User Groups” from transactional log data. There are numerous steps necessary to complete this goal. These steps range from storing the logs, converting them into a usable format, clustering the data extracted from the logs and deriving groups from that data.

Before explaining in detail all of the steps required to complete the goal of this project the pipeline, must be defined in full. This will constitute an overview for the design of the project as a whole.

3.1.1 Overview of the Pipeline

The pipeline, as previously stated, is a series of operations performed on the inputted transactional log data with the intention of producing data that can help extract user groups for the users contained within the data. This pipeline can be divided into three main stages.

The first stage is called the “Extraction stage". At this stage, the input log data is extracted from the logs and converted into a format that is both readable and usable for the secondary extraction process. The “User Features” are then extracted from this data.

The second stage is called the “Clustering Stage". At this stage, all of the “User features” extracted in the previous stage are brought together to form clusters. These form the basis of the “User Groups".

The third and final stage of the pipeline, is the “Validation stage". At this stage, the clusters created in the previous stage are analysed to determine both their stability and to see what they represent.

Upon the completion of the third stage of the pipeline, the outputted clusters are used to form “User Groups". An overall view of the pipeline is shown in figure **X**

3.2 The Extraction Stage

This is the explanation for the design of the first stage of the pipeline called the “Extraction Stage". It is so called because a core tenet of this stage is the extraction of usable data from given sources. The stage can be divided into smaller sections. Each of these “substages" represent an operation that must be completed to continue the process of deriving “User Groups” from the transactional log data.

These substages are:

1. Extracting Data from the Logs.
2. Defining a User.
3. Extracting the User Features.

3.2.1 Extracting Data from the Logs

To extract data from the logs it is necessary to fully understand both the type of logs in use and the nature of the data to be extracted. To begin, one needs to describe the type of logs that are in use in this project as well as their background. One must then describe the nature of the data contained within the logs as well as why the data should be extracted and into what form.

Background to the Dataset

The transactional logs in use in this project are records of interactions between users and systems. In the case of this project the system in question is a web based Information Retrieval system. This system is an archival domain. An archival domain refers to an online archive of historical documents. This archive has web search logs that show user interaction with the archive. This is in the form of user queries or in the users' "clicks" on artefacts of the web archive. This archives web search is not parametrised to learn from the user or to develop any form of preferential results based on past searches. All words that are entered are searched for within the database using only those words.

The domain in question belongs to the University of Dublin, Trinity College. In particular they belong to the Digital Collections of the Library of the University of Dublin, Trinity College. This collection is located at *http://digitalcollections.tcd.ie*. The logs in question were collected between the 24th of January 2016 and the 25th of January 2017, roughly a year of log data.

This website contains the archival data of the large collection of historical artefacts (such as photographs, letters etc.) belonging to the university. All of these are marked with a name and ID that is contextual to the artefact. This is done using Encoded Archival Description (EAD) (Dublin, 2016). EAD is an XML standard for encoding archival finding aids. These aids are used by researchers to determine whether or not the archive collection contains materials relevant to their research (Stockting, 2004). The users are searching for this data.

As it is an archival domain, a large portion of the userbase is academia. They are likely to be either researchers, lecturers or students. However, it should be noted that the draw of unique works such as the Book of Kells, means that their are non-insignificant portions of non academic users in this dataset. It is the unique nature of this archive that allows for the project to proceed. The userbase is limited to those with either an academic or personal interest in historical artefacts. The limited nature of the users it receives permits grouping the users based on their features.

The Transactional Logs

Log files are a record of the transactions that take place between a user and a system. They contain knowledge such as which user accessed what resource. Logs are stored in plain text format usually with a ".log" filename extension. This format minimises dependencies on system processes, which allows for logging when those processes are unavailable.

As the system is a web based archive it naturally has web servers. There are two web servers of interest to this project with their own log type in use. These are:

- **Query Logs:** These are the logs of Apache Solr web server. It handles the search system used by the Digital Collections website. The logs here containing the queries the user used to search the system. An example of a log entry is listed below.

  ```
  0.0.0.0 - - [01/Jan/2016:00:00:01 +0000]
  "GET /home/js/briefDoc.js HTTP/1.1" 200 1210
  "http://digitalcollections.tcd.ie/home/
  index.php?dris_id=ms58_003v"
  "Mozilla/5.0 (compatible; MSIE 9.0;
  Windows NT 6.1;Trident/5.0;  Trident/5.0)"
  ```

- **Server Logs:** These are the logs for the Apache web server that handles the interactions the user had with the server post query. These interactions are the clicks the user had on the web artefacts displayed as the result of their query. An example of a log entry is listed below.

```
1.1.1.1 - - [01/Apr/2016:00:00:01 +0000]
"GET /solr/dris/select?rows=0&facet=true&
facet.limit=1facet.mincount=1&json.nl=map
&facet.field=title&facet.field=name&
facet.field=PID&facet.field=shelfLocator&q
=*:*&wt=json&json.wrf=jQuery
17204357232106849551_1461715900840&_
=1461715903400
HTTP/1.1" 200 1204298IE 9.0; Windows NT 6.1;
Trident/5.0; Trident/5.0)"
```

Both are formatted in Apache server format. This format, also known as “The Common Log Format", is the standard text file format used by web servers when generating server log files. This standardisation lends itself to analysis and information extraction using a variety of technologies.

The structure of the the logs is such that each contains an IP address, a timestamp and a GET request. There are other sections contained within each log entry but, other than these, none pertinent to the project.

3.2.2 Defining a User

Having explained the nature of the dataset it is necessary to define a “user” in the context of this project. A user is any individual who has interacted with the Digital Collections system in the time between January 2016 and January 2017 and whose interaction is recorded in the log data.

However, there are issues with leaving the definition so fluid. There are billions of individual interactions recorded within the logs. If a definition of what was accomplished by different users is not defined, then all of those interactions will not be capable of being grouped meaningfully.

One possibility is to determine each IP address as recorded within the logs as being an individual user. While this solves the proceeding problem of too many users that hold data of no relevance, it may create another problem. What if there are multiple users using the same IP address? Some users of the system may be doing so from common workspaces. As such, it is likely that their IP addresses would be the same. The data extracted would therefore be conflicting as each user may end up representing multiple people who may belong in different "User Groups". This would make the "User Groups" inaccurate and so of little value.

Instead, it is better to define a user as being an IP address that interacted with the system within a time period dating from the first interaction. By doing so the previous two problems are avoided. There is still a possibility of multiple users from the same IP address being in conflict and being recorded as one user. However, by limiting the time period with within which the "user" is defined, the likelihood of this problem occurring is minimised. The drawback is that the interactions the user has with the system over the full span of time the dataset covers is not extracted. While this would be interesting to know, it would be less beneficial comparatively.

The period of one hour after a Users first interaction with the system, will define what a “User” is. This time period will be referred to as a “Session".

Each session is defined as being the period of one hour after a user's initial interaction with the system. In the context of the logs that first interaction will come in the form of a search query contained within the SOLR logs. This is because the majority of users are assumed to have used the search system to gain access to the archives artefacts. It must be noted however, that some users may have entered the system via third party links.

Within a session each further interaction recorded from that user(shown by IP address) will be counted as being done by the same user as opposed to any different user. An interaction by a user with the same IP address done outside of that session time will initiate a new session. It should also be noted that any interaction towards the end of a session performed by the user, will extend the session duration by fifteen minutes. Should more interactions by the same user be found in this extended time period, the session will continue to be extended as previously until this is no longer the case. In this way, the session will hopefully remain accurate to the user it is representing.

There is a reason why the length of the session is initially capped at one hour, as opposed to the session duration of 30 minutes used in *in Search Log Analysis of User Stereotypes, Information Seeking Behavior, and Contextual Evaluation*(Kamps and Zhang, 2010). In their paper it was assumed that 30 minutes was the most a user was likely to spend querying and interacting with the system. However, in making that assumption they do not account for the users who may take breaks in the middle of their sessions and then return. By failing to account for this they expose themselves to the possibility of double counting users. Due to the nature of their project such considerations were not essential to the project as they were broadly stereotyping users, not grouping them. However, in this project such double counting could affect the accuracy of the resulting data.

3.2.3 Defining The User Features

Having defined the user and the session as well as giving a background to the dataset, it is now necessary to define the "User Features" that will be extracted. These are the features to be used to cluster the users in the clustering stage. These features must meet certain criteria. These criteria include being clusterable, being replicable across all of the users (or most of them) and that they return information relevant to the users.

The User Features chosen were chosen in part because in (Kamps and Zhang, 2010) such features were used to successfully sort the users into into two separate stereotypes. They have proven usefulness in a project that attempted a similar concept using a similar dataset. It should be noted that all of these features are defined as being on a per session basis as it is the users of each individual session that will be clustering.

The user features in use in this experiment were:

- **Session Duration**
- **Average Query Length**
- **Number of Repeated Queries**

- **Number of Total Queries**
- **Dwell Time**

Briefly, these terms mean;

Session Duration is the length of the session in time. This is useful because it can be quantified as a number. This makes it easier to cluster. Furthermore, it can be assumed that users of the system with very short session durations are users with little knowledge of the domain. This can be assumed because a user who actually has an understanding of the domain in question is more likely to stay with the system for a longer period of time so as to retrieve which they are seeking (Kamps and Zhang, 2010).

It was calculated according to the equation 3.1.

$$\text{Session Duration} = \text{Timestamp of Final Interaction} - \text{Timestamp of Initial Que} \tag{3.1}$$

This result was given in seconds, and it is this value that was used as a user feature.

Average Query Length is the average length of the queries used whilst searching the archive. This is measured in number of terms as opposed to number of characters so as to give meaningful comparison between users. It was computed according to the equation in 3.2.

$$\text{Average Query Length} = \frac{\text{Total Length of all Queries}}{\text{The number of Queries}} \tag{3.2}$$

This was selected because the assumption is the shorter a query is means the more general the query is. A longer query will typify a more specific information need which in general represents a more knowledgeable user. (Kamps and Zhang, 2010)

Number of Repeated Queries is the number of queries repeated whilst searching the archive. This was chosen because the larger the number of repeated queries then the more likely a user was unable to retrieve the information they wanted initially. So, a smaller number of queries will indicate a user that was satisfied faster.(Kamps and Zhang, 2010)

Total Number of Queries is the total number of queries used in a session by a user whilst searching the archive. This was chosen because larger the number of total queries the greater the engagement a user had with the system. This is particularly useful when used in conjunction with the number of repeated queries to give a likelihood of user satisfaction of the system and knowledge of the domain.(Kamps and Zhang, 2010)

Dwell Time is the duration of time a user spent interacting with the system post query. This could also be described as the amount of time a user spent interacting with the results of their query. This was chosen because the greater the dwell time the greater the likelihood the user was to have been satisfied with the results retrieved and the greater the likelihood the user had a higher understanding of the domain (Kamps and Zhang, 2010).Dwell time was calculated according to the equation 3.3.

$$\begin{aligned}\text{Dwell Time} = \text{Timestamp of Final Apache Interaction} - \\ \text{Timestamp of Initial Apache Interaction}\end{aligned} \tag{3.3}$$

This value in seconds was what was used as the value for clustering purposes for this feature.

3.3 The Clustering Stage

In the clustering stage, the user features are clustered using the K-means algorithm to form clusters. These clusters are then in turn validated and analysed in the validation stage. As the clustering process has already been explained in chapter 2 therefore it is unnecessary to explain clustering process here.

The clustering in question will cluster sessions based on user features. This will hopefully form clusters of users with similar user features from which meaningful user groups can be formed.

3.4 The Validation Stage

This stage is the final pipeline stage. It is called the “Validation Stage". Here, the clusters are validated and analysed. The process of validation uses Silhouette scores to determine cluster validity. As Silhouette scores have been explained previously in chapter 3 they will not be explained again here as it would be unnecessary.

The purpose of this stage is to analyse the clusters to ensure that the data that is clustered can have user groups derived from it.

Chapter 4

Implementation

4.1 Introduction

The purpose of this project is the idea of creating a replicable pipeline through which transactional log data could be converted into "User Groups". In order to accomplish this a test environment in which the pipeline could be built and tested has to be established. Furthermore, the pipeline itself had to be implemented according to the specifications laid out in chapter 3.

There had to be an implementation of all 3 stages of the pipeline (Extraction, Clustering, Validation). There also had to be methods to evaluate the given clusters beyond their validity as clusters so as to correctly derive User Groups from that data.

It is necessary to review the tools used to implement this project, followed by a brief description of the test environment of the project. This will then be followed by the implementation of the pipeline outlined in chapter 3, together with a discussion of how results were derived from the pipeline. Finally the issues encountered in the course of implementation of this project are outlined.

4.1.1 Tools in Use in this Project

In this project several tools and libraries were used to effectively implement the project according to its design.

The transactional logs used in this project were stored, parsed and searched using two tools from the Elastic Stack. This is a suite of open source tools from Elastic designed to help users take data from any form of source and in any format and search, analyse, and visualize that data in real time. Whilst there are several tools in this suite the only tools in use in this project were Logstash and Elasticsearch.

Logstash is an open source, server-side data processing pipeline that inputs data from a source, transforms it, and outputs it to a chosen destination. It acts as a log aggregator that can perform filter actions. It is used in this project for collecting and parsing logs. It accomplishes this through configurable inputs such as socket/packet communication and file tailing.

Elasticsearch is a distributed, RESTful search and analytics engine which is written in Java. It is built upon Apache Lucene and has the ability to handle a large number of search requests. Using HTTP/JSON protocol it takes data and optimizes the data according to language based searches. It then centrally stores the resulting data. An instance of Elasticsearch on a server is called a cluster. Within that cluster, indices are used to store documents, which is data that Elasticsearch can search through and alter.

The main project consisted of several Python scripts which when run, executed the pipeline from beginning to end. These scripts were written using Python 3.6.1. These Python scripts, relied on several Python libraries to implement the pipeline. The two most important of these were the Elasticsearch Python Library and scikit-learn.

Elasticsearch is a Python interface to the Elasticsearch api which allows for Python scripts to interact with Elasticsearch the same way that a user would. This was used to allow the Python script to interact with the log files in real time.

Scikit-learn, is a machine learning library for the Python language. It contains both supervised and unsupervised learning algorithms that include regression, classification and clustering algorithms Pedregosa et al., 2011. In this project, it provided both the clustering and cluster validation tools.

4.2 The Test Environment

The environment that this project was developed on was a Virtual Machine located within the School of Computer Science and Statisticsof the University of Dublin, Trinity College. This virtual machine ran an instance of Debian version 8. It had 4 GB of ram as well as 64 GB of memory. The transactional logs in use in this project were stored in a plain-text unencrypted format on this virtual machine.

This machine had only three users. An SCSS network administrator, a profile for the researchers assisting the project, and the author. Access was strictly regulated and password protected. It was not possible to gain access to the virtual machine when one was not connected to the internet network of the university. As such, protections were in place to prevent the leak of any confidential data.

4.3 Configuring The Pipeline

Prior to the three stages of the pipeline being implemented, it is necessary to convert the transactional logs from a log format to a form that allows the data contained within to be easily extracted. The format log files are written in is called the "common log format".

This format consists of the IP address of the user interacting with the server of the log in question, followed by a timestamp. This is followed by the GET request generated by the interaction. This is the actual interaction performed by the user in a form understood to the server. There are other parts of each log entry but for the purposes of this project they are superfluous.

Due to the high amounts of noise between interactions due to images and JavaScript files being loaded it was necessary to parse the logs. To that end the logs are parsed and then entered into a document searching and storing tool. This is done first with Logstash and then Elasticsearch.

4.3.1 Parsing and Piping the Logs using Logstash

Logstash is a tool used for the parsing and piping of data which does this using configuration files. These files take in an input, run operations on that input and then output them to a chosen destination. In this project the logs are used as an input and then they are put through a "grok filter".

The grok filter is a plugin for Logstash that parses log files according to set patterns. Although the plugin comes with many inbuilt patterns, for any log file that does not meet one of those patterns' standards it is necessary to create a new pattern parser. The Apache weblog format is common, as such the grok plugin comes with an inbuilt parser for it. However, the SOLR logs have a slightly different form of the common log format and as such they require a unique grok parse to successfully parse them.

Upon being parsed each individual log file was split into separate "documents". Each of these "documents" represents an interaction from an IP address to the server in question. This includes the timestamp and GET request also.

On finishing the parsing the decision of where to send the logs needs to be made. In the case of this project, as they are tools configured to run together natively, the decision was to send the logs to an Elasticsearch cluster set up on the same server.

Due to having two different log types (Apache and SOLR), it was necessary to create a particular configuration file, so that it would parse the logs separately and then output them separately to different destinations within the Elasticsearch cluster.

4.3.2 Storing and Searching through the Logs Using Elasticsearch

Elasticsearch is a tool used for the storage and searching through, or querying of data. In the case of this project the data in question is the stored transactional logs that have been parsed by Logstash into "documents".

These “documents” are then stored in one of two different "indices" on the cluster, depending on their log type. The two indices were "apache_index" for the Apache web logs, and "solr_index" for the SOLR logs. On the completion of the parsing by Logstash and subsequent storage in Elasticsearch the indices had the following number of "Documents" each 4.1.

TABLE 4.1: Table of Elasticsearch Storage

Index	Number of Documents	Size of Index
solr_index	545963	332.7mb
apache_index	25995545	12.3gb

As each document represents an interaction by user with the server in question there is clearly a large amount of data to be parsed and then extracted.

4.4 The Extraction Stage

This primary stage of the pipeline was dealt with entirely using Elasticsearch, and its Python api library. It was used first to extract the relevant data from each "query" or "document"(user/server interaction), and then the individual features were extracted and calculated from that data.

4.4.1 Extracting the Data from the Logs

Having stored the logs in their document form in Elasticsearch, the relevant data was extracted in the form of user sessions. These sessions as explained in the chapter 3 are what will be used as representations of the individual users. As such, it is necessary to extract the individual sessions, prior to extracting the features for each of those sessions.

The sessions were extracted using the Elasticsearch Python library. This library implements the functions that Elasticsearch has in a format usable by Python. To extract the sessions, the script searched for an interaction from any IP address to the SOLR logs. Then, it took that IP and performed a second search to return all interactions that IP had with the server between the timestamps from the initial interaction, and the interaction one hour from that point. If there were interactions within that hour time frame to the SOLR server from the same IP address, the ending timestamp was readjusted to fifteen minutes beyond the final one of those interactions' timestamps.

Below is a sample Python Elasticsearch, search function that returns all of queries of every IP address that matches the $user$ string.

```
es.search("solr_index",
{"query":{
        "bool":{
            "must":[{
                    "match":{"clientip": user}
             },]
       ,}
     }})
```

These alterations to the timestamps were calculated using the Python datetime library's timedelta function. First the timestamps were converted into datetimes from strings, and then the timedelta was applied to them. The timedelta in this case was the difference in time between the timestamp that was begun with and the new timestamp. In this case of this project, it would either be one hour for creating the session or fifteen minutes for the sessions' extensions.

Upon creating the session using the data from the SOLR log, the initial and final timestamp were then used as parameters, along with the IP address of the user of the session in question. These parameters were then used to create a new search through the Apache web logs. This search returned all of the interactions that that IP address had with the Apache server between the initial and final timestamp.

With these methods the data from the logs was extracted and then parsed into sessions. When each session was extracted, the user features of those sessions were calculated.

4.4.2 Extracting the User Features

To extract each of the user features, they had to be calculated individually on a per session basis. Each of the user features were calculated to reflect the design laid out in chapter 3.

Upon being calculated the User features were added to a dictionary. In this dictionary, the value was a tuple of was the IP address, followed by the five user features. The key was a session identification number that was unique to each session.

The user features that were calculated were:

- **Session Duration**
  The length of each session from the initial timestamp to the end one.
- **Average Query Length**
  The average length of each of the queries in each session. Where length is the number of terms in each query.
- **Number of Repeated Queries**
  This is the total number of times a query was repeated in each session.
- **Number of Total Queries**
  This is the total number of queries that were used in each session.
- **Dwell Time**
  The length of time that a user spent interacting with the result of each query.

4.4.3 Session Duration

Session duration, is the length of the session in total. It was calculated by getting a timedelta two timestamps. The first, is the timestamp of the first interaction in the session. This is taken from the timestamp of the SOLR log interaction, that starts the session.

The second, was taken from the final interaction in the session. This could be on of two values. In the case of there being further interactions by the user within the session time frame in the Apache logs, the final of those timestamps, was used. However, in the event that there were no interactions post query with the query results in the time frame, then the timestamp of the last query, or SOLR interaction was used.

This was result was calculated by calculating the timedelta between the two timestamps in question. First they were converted to datetimes. Then the timedelta was calculated from that value. This is according to the formula shown in chapter 3.

4.4.4 Average Query Length

The average query length of each session was calculated very simply. Upon the extraction of each query from the session, the query string was parsed according to the number of blank spaces. In doing so they were separated into a list of terms. The length of this list was the length of the query in terms.

This number was then added together with the length of each and every other query in that session. Again all of the lengths were in terms. A count was started when the first query was extracted. Therefore upon the cessation of the session, both the total number of queries, and the total length of all queries cumulatively were found. With these the average query length was calculated according to the formula shown in chapter 3.

4.4.5 Number of Repeated Queries

To calculate the number of repeated queries was similarly simple. I created a list of queries for each session. Upon a query being extracted, it was compared to every query in that list. This was done using Pythons inbuilt string comparison function. If there was no match, that query was added to the list. If there was a match however, a counter was incremented. This counter is what was used as a value for the purposes of clustering this user feature.

4.4.6 Number of Total Queries

To calculate the number of total queries, a counter was created a that was incremented upon the extraction of each query for that session. This counter was used as as a value for the purposes of clustering this user feature.

4.4.7 Dwell Time

Dwell time was calculated using the extracted data from the Apache logs. The interactions for the session were taken from those logs, as explained above. Similarly to calculating the session, the first timestamp for the interactions of the session was parsed and converted into a datetime. The timestamp of the final extracted interaction was then parsed and converted into a datetime. A timedelta was then calculated based off the difference of those two datetimes, according to the equation in chapter 3. This value in seconds was what was used as the value for clustering purposes for this feature.

4.5 The Clustering Stage

Upon completion of the extraction stage of the pipeline, the features as well as their identifying IP addresses, and session ID were clustered. This clustering was K-means clustering, as explained in chapter 3 and chapter 2. This K-means clustering was done using the Python machine learning library "scikit-learn". However for that to be done the results had to be converted from their tupled form into a form usable by scikit-learn.

To that end upon completing the extraction of all interactions, and the formation of all sessions, the resulting dictionary was parsed. Each of the relevant elements of the values tuple, was converted into a list of that particular sub-value. These sub-values were the IP address of the user, as well as all five of the user features. These lists were what were used to create the clusters.

4.5.1 Clustering the Users based on their Features

Due to the number of features in use several different clustering methods were used to gain different insights into the interactions between different user features. While the overall method of those clusters was the same (i.e K-Means), each of the methods took in different types and numbers of values numbers. The value types were the features, and the number was from the minimum required to cluster(two) all the way up to the maximum number of features(five).

As the overall clustering method is the same for each of these methods, only the general clustering method will be described, without making reference to any particular type or number of features.

The lists for the features to be clustered, as well as the list of the users, are taken into the clustering function. There the feature lists are converted using the numpy transform method, into a multi-dimensional array, where the number of dimensions is equal to the number of features. This array is what is taken into the scikit-learn k-means clustering function.

At this stage it is necessary to select the number of clusters for the data to be clustered into. Having done so, as well as transforming the data into a suitable form for the method, the k-means fit_transform() method is invoked. This method computes the clustering as well as transforms the results into cluster-distance space. This can then be graphed to represent the results of the clustering visually.

Immediately after doing also the predictions for the cluster were computed. This is done using the predict() function. This is a function that will predict which cluster is closest for a given sample set of values. Upon giving it the full set of values, the results, are of the same size as the number of each individual feature, as well as the number of sessions and thus the number of users. They also maintain the same order as before. As such,it is possible to map which user is closer to which cluster, based off of the results of this function.

4.5.2 Graphing the Results

Having clustered the features, it is necessary to display these clusters in such a way as to be visually interpretable. This allows for a clear representation of any fault in the system. To that end two separate diagrams were used to display the results of the clustering. The first is a scatter plot graph, and the second a cluster regions diagram.

Scatter Plot Graphing

The scatter plot graphing was done using the matplotlib Hunter, 2007 library. Using its inbuilt plot function all of the clusters data points were plotted onto a graph. If there were only two features in the cluster then the axis were the features in question.

However if there were more than two features, the diagram could not be drawn due to the multidimensional nature of the data to be graphed. It was therefore necessary to transform the data into a form that could be plotted. This was done using the PCA function of scikit-learn.

PCA stands for Principal component analysis. This is a linear dimensionality reduction that uses singular value decomposition of the data in question, to project said data to a lower dimensional space. By using this it becomes possible to graph data that is beyond two dimensional.

Having plotted the points, the cluster_centers_ value that is returned by the k-means fit() function was used. This is an array that contains the locations of all of the cluster centres, or centroids. These were marked with an x. This allowed for clear viewing of the centroids against the clusters.

Cluster Region Diagrams

The cluster regions diagrams, are diagrams that are also generated using matplotlib Hunter, 2007. These diagrams, show each of the clusters regions. These regions are areas of the graph, where the points within are part of that cluster. They differentiate each of the voronoi cells of the graph by colour. Each of these cells is a representation of a cluster region. Similarly to above, PCA was used for higher dimensional data, and the centroids were marked with a large x so as to be found with ease.

4.6 The Validation Stage

Upon the successful creation of the clusters, the final stage of the pipeline is entered. Here the validity of the clusters is determined according to their silhouette scores. This is according to the design in chapter 3 and using the method explained in chapter 2. The silhouette scores in question are calculated using the Python machine learning library “scikit-learn”.

4.6.1 Determining Cluster Validity

To determine the silhouette score of each of the clusters the silhouette metric function was used. This function takes in several arguments. The first argument is the data to be checked, followed by the predicted labels of that data. These labels were taken directly from the results of the clustering using the .labels_() method.

The next argument is the method with which to calculate the distances between instances of the given array. For this euclidean distance was chosen as it is with this that the clusters are calculated. Finally the function takes in an the sample size of the inputted data to check. Due to the size of the data, it was not possible to use the full clustered data as a sample size. As such the largest possible sample sizen was used. In this case, it was 1/4 the size of the inputted data.

Some of the resulting silhouette scores for the different cluster types can be found in 4.2

TABLE 4.2: Table of Silhouette Scores for the User Features of in the Unaltered Paramter

Number Of Clusters	Silhouette Score
2	0.652947835827
3	0.606859519945
4	0.602480831205
5	0.603267673385
6	0.633834602068
7	0.744855455533
8	0.797511097467
9	0.832294378563
10	0.91636005137

4.7 Cluster Preprocessing

In order to achieve clusters that made data trends more evident than the initial version of the dataset, certain preprocessing was done to the user features values that are used to create those clusters. This preprocessing created several parameters. These included getting the natural logarithm of the feature values, filtering the dataset as a preprocessing action and naturalising the feature values.

4.7.1 Getting the Natural Logarithm of the Features

Getting then natural logarithm of data points, is a method of reducing noise in a dataset. During the course of the implementation of this project, it was observed that the data being graphed by the logs was noisy. Therefore, a second dictionary of features was created. In this case, all of the features in question have values that are the natural logarithm of their normal values.

This was achieved by gathering all of the values as per normal, except when it came to adding the features to the dictionary of the features, that was stores the feature values. There a second dictionary was created that held a version of the data with a natural logarithm function applied to each features' value. This was done using the verblog() function of Python.

This is a function that takes in two arguments normally. They are the value of the number to be get the logarithm of, and the value of the logarithm for it to be put to. However, if no second argument is inserted, then it applies the natural logarithm.

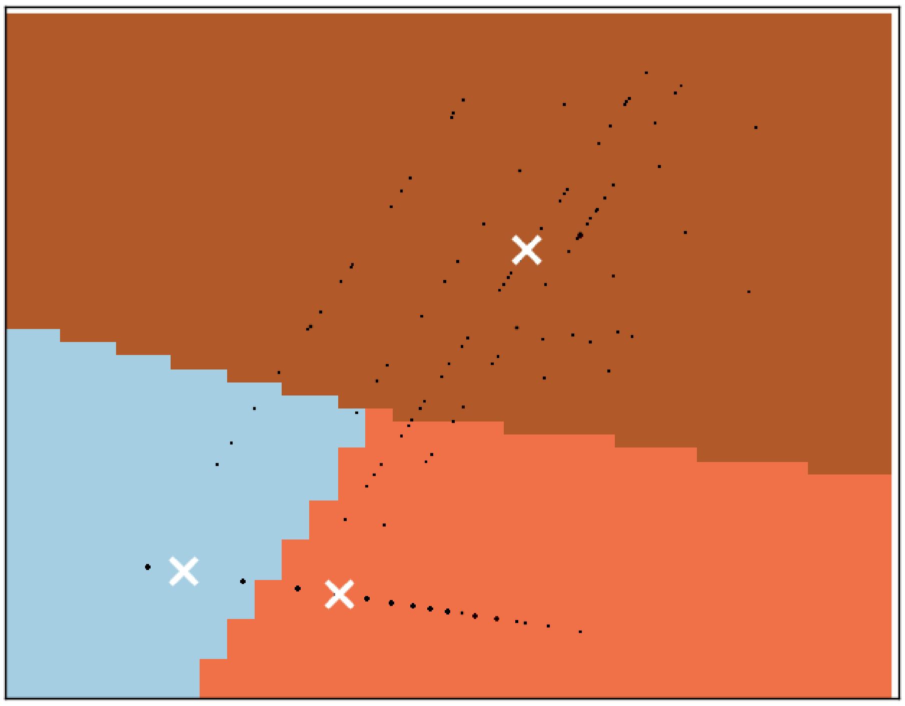

FIGURE 4.1: Average Query Length, No. of Repeated Queries and Dwell Time of the Logarithmed Parameter clustered into 3 Clusters

An example result of this attempt can be seen in figure 4.1

4.7.2 Dataset Preprocessing

Another method implemented was to reduce the dataset size. While this is not advisable, as it can skew the data uncontrollably, by choosing how the data that was excluded was excluded, it was hoped to skew the data into a manner that made the data easier to cluster.

To that end all sessions that contained a dwell time of 0 were excluded. This was done by creating another dictionary, similar to the previous one. In the event that a sessions' dwell time was zero, that session was excluded from this dictionary.

In this way it was hoped to eliminate those users who had arrived at the website by accident. This may have reduced the linearity of the dataset, by removing the lowest end of the users, in terms of feature value. This would thus leave just the users with generally higher feature score, that might yield more interesting results.

Some of the results of this can be seen in figure 5.2.

4.7.3 Normalising the Feature Values

Having attempted two other forms of cluster preprocessing, next the values of the user features were normalised. Normalisation, is the scaling of values on different scales to a notionally common scale. In the case of this project all feature values were scaled to be between 0 and 1. This was accomplished by applying equation 4.2 to the feature values on extraction.

$$z_i = \frac{x_i - \min x}{\max x - \min x} \tag{4.1}$$

Where z_i is the i^{th} normalised data and x is the set containing all users individual feature values for that feature.

Normalised versions of the three previous datasets (the initial dataset, the logarithmed dataset, and the Preprocessed dataset) were created by adding normalised versions of their feature values to a dictionary at the same point as they were added to their dictionaries.

This was done in order to make the outputted graphs and values clearer by putting all the data values on the same scale. Some of the results of this can be seen in figure 4.2:

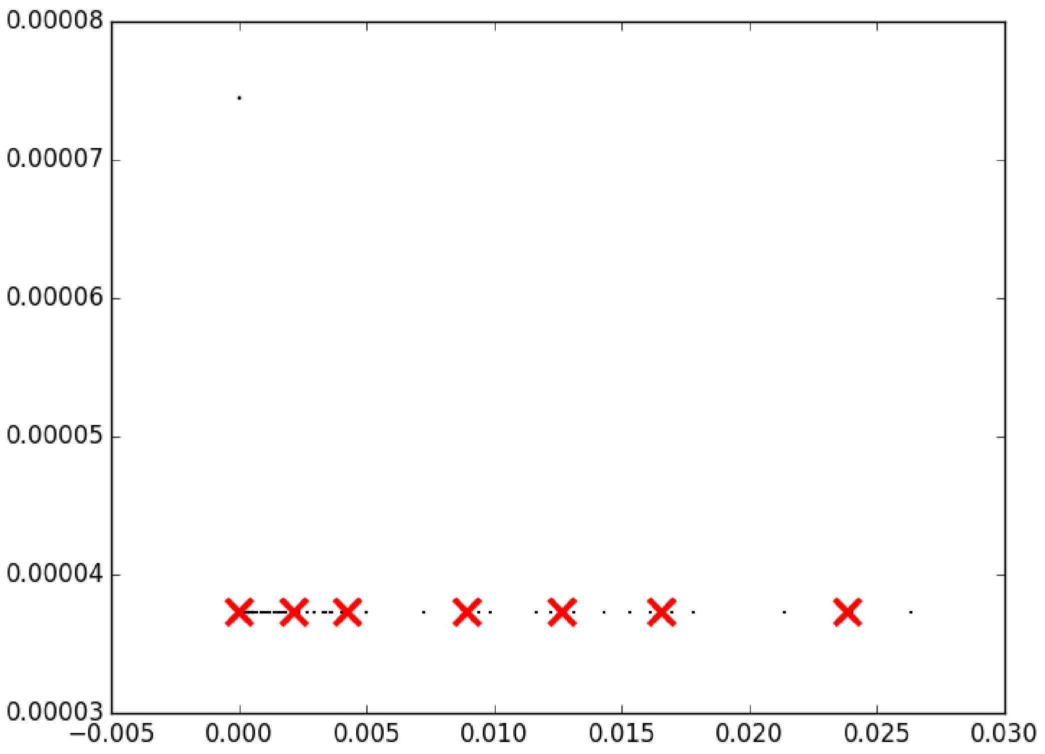

FIGURE 4.2: Dwell Time and No. of Repeated Queries of the Normalised and otherwise Unaltered Parameter clustered into 7 Clusters

In summary, at the end of the preprocessing there are 6 distinct versions of the dataset. They are the unaltered, the logarithmed, the filtered and the normalised versions of each of those. Each of these datasets are taken into the clusterings stage as a dictionary composed of a session identification number as a key and an IP address, and a user feature value for each of the features tupled together as a value. Each dictionary was of the same size as the number of sessions extracted.

4.8 Implementation Issues

Throughout the implementation of this project, issues arose. While some were minor and unworthy of mention, some others were to a degree that made executing the project far more difficult than would be initially presumed. These can be roughly divided into two separate issue areas. These are: memory issues and data issues.

4.8.1 Memory Issues

Due to the extremely large nature of the dataset in use, as well as the size of some of the feature values, it was possible to run into memory issues whilst performing clustering or related tasks. This occurred in part due to the need for Elasticsearch to be running during the course of the clustering. Elasticsearch as a tool was quite resource intensive due to the size of the dataset it was storing and querying.

As the issue was due to the nature of the tools in use it was not possible to fix the underlying root cause of the issue. Instead alterations were made to the Python scripts at the point that the issue would occur. By altering parameters such that the data that was causing the issue was treated differently to the rest of the data at those points, the issue was overcome.

4.8.2 Data Issues

As will be seen in chapter 5 issues occurred in the post clustering analysis of the clusters. Whilst there was no issue with the validity of the clusters, the graphical output of the clusters showed that the clusters had not formed according to the expectations.

The assumption upon beginning the project, was that the data would yield clusters that were of the normal K-means variety. These being tight clumps of data points that would be roughly spherical in shape. Instead the resulting graphs showed data that was either extremely noisy, or extremely linear in shape. Often the results were both linear and noisy.

Chapter 5

Evaluation

5.1 Introduction

In total, upon running them projects Python script on the dataset, 643 files were generated per parameter. The parameters in this case refer to the six variants of the dataset I ran my program against.

The first three parameters were the normal full dataset of users, a dataset where all of the user features values were substituted for the logarithm of those values and a test dataset which removed all users without a dwell time below a certain threshold. The other three were the same datasets as above, except that their feature values were normalised to be between 0 and 1.

At the conclusion of the program that was run, 3858 unique files had been created. The large number of files generated, is in part due to the six different parameters, it is also however due to the repeated clustering of features with different numbers of clusters (between 2 and 10). It is also due in part to the need to output all results to a file in order to view them. Due to the remote nature of the test environment if any result was to be viewed (e,g predictions for which cluster a user was in for a particular parameter and feature set, etc.) it needed to be outputted to a unique file that could be analysed.

All of these files contain pertinent information regarding the projects outcome. However, due their size, the vast majority of the results will not be included in this document. Results that typify the issues that arose over the course of this project will be displayed and discussed. Some of the results, that are not displayed in this section, are within appendix A.

5.2 Results of the Clustering

There are three main forms of cluster graphs whose information is pertinent to the analysis of the clusters.

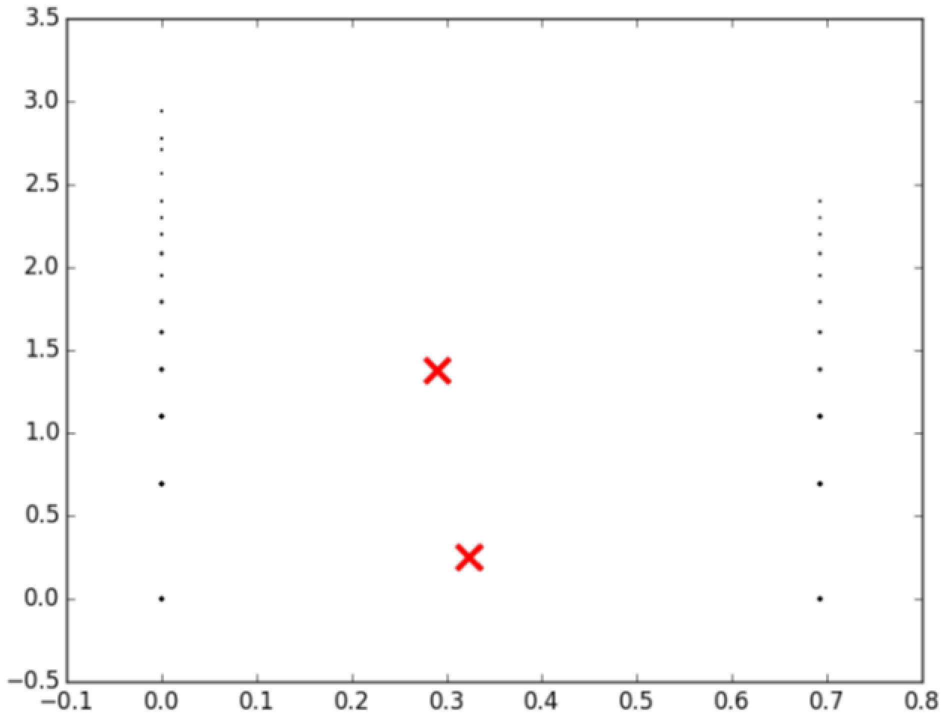

FIGURE 5.1: Average Query Length and the Number of Repeated Queries in a Session Clustered into Two Clusters

Figure 5.1 is a scatter plot diagram that shows clusters of the unaltered parameter that has been clustered into two clusters.

The features used were average query length and number of repeated queries. It is evident from a first glance that this graph does not display a useful K-means clustering result.

There are no typical spherical clusters formed. Instead there are two parallel and linear groups of data. Each of those lines represents a feature. The resulting data points are extremely sparse also. Beyond that, the Silhouette sore for these clusters is 0.9657. This high Silhouette score demonstrates a key failing of Silhouettes, which is that if the logarithm in any way is unsuitable for the data (or vice versa) it can not cope with this.

The lack of normalisation may have played a small role in the outcome of the distance between the two groups of data. However the linearity of the graph could not be altered by normalisation.

Figure 5.2 is a cluster region diagram that shows the voronoi cells that clusters of the filtered parameter that has been clustered into eight clusters. All 5 features are clustered together in this graph.

This graph displays less data than a similar graph(such as 5.3). This is clear evidence of the filtering removing users that is skewing the data. However this graph also displays the sparsity that is the result of such an action. Little meaning can be derived from this graph, due to its low Silhouette score (0.5674), which shows that the clusters that have formed, are inaccurate.

Figure 5.3 shows a cluster region diagram that shows the voronoi cells that clusters of the logarithmed parameter that has

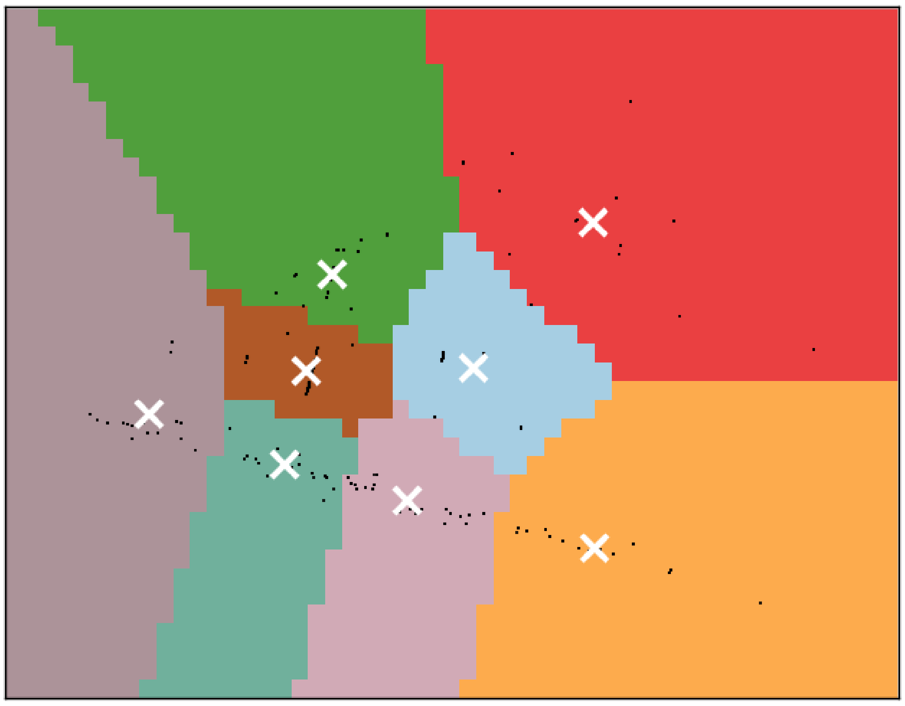

FIGURE 5.2: All Five Features of the Filtered Parameter clustered into Eight Clusters

been clustered into ten clusters. All 5 features are clustered together in this graph.

This graph displays more data than the proceeding two, due to the higher number of features, and the lack of filtering. However the trends that were expected to be shown with the logarithmed data have not appeared.

Moreover, this graph is both linear, noisy and poorly clustered. Its silhouette score is 0.4432. The large black bars are groups of linear data points grouped together.

It can be seen from these graphs that the clustering has been

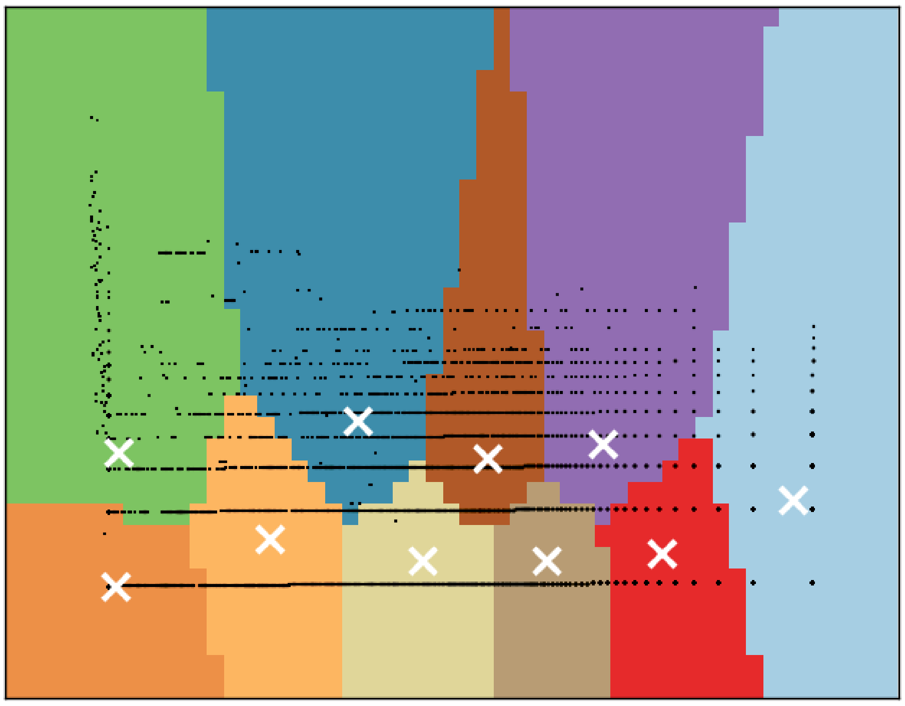

FIGURE 5.3: All Five Features of the Lograithmed Parameter Clustered into Ten Clusters

successful, but that no meaning can be derived from these clusters. This can be assumed to be from several different causes.

K-means does not handle non spherical clumped data well. Evidently the data at hand is not of a type that K-means is able to cluster effectively. Alternatively, different, more clusterable user features could have been chosen that may have been better able to form clusters.

Chapter 6

Conclusions

The main goal of this project was to develop a pipeline which would take transactional log data as input and then output data clusters from which user groups could be derived. The solution proposed in this paper, involved the parsing of logs into separate interactions using Logstash and Elasticsearch. These logs were then to be searched through to form sessions. Using user features inspired by Kamps and Zhang, 2010, these sessions were to be clustered using scikit-learns implementation of the K-means algorithm. The formed clusters were to then be validated using sci-kit learns implementation of Silhouette scoring. The resulting values could then be taken as validation of the clusters which could be further analysed to derive the user groups.

This project has not succeeded in its attempt to build such a successful pipeline. While the author remains confident in the general design and idea of the pipeline, the failure of this project to successfully derive user groups means that the question of where the failure stems from must be answered.

It is the opinion of the author, that the failure to form clusters of a stable nature stems from one of two sources. These are either the algorithm chosen to cluster the data, or the data itself.

If one takes the view that the overall design of the project is sound, then the failure of the project can be seen to stem from the data itself. The data was found to be noisy and linear in nature. Neither of these are aspects of data that K-means deals with effectively.

Alternatively, one could take the view that the projects goal was to form user groups from the given data. If this is the case, then the failure results entirely from the choice of K-means as an algorithm. The author feels that to assume such is incorrect, as it places to great an emphasis on the choice of algorithm as opposed to the inherent problems with the datasets clusterability.

The projects goal was to create a replicable pipeline through which transactional log data could be converted into user groups. While user groups were not derived, clusters based on user features were. The inability of these clusters to yield user groups can not diminish this fact.

This project as an experiment has failed, but a failure is still a valid result and one that can be learned from. Going forward, this experiment could be repeated using different datasets or algorithms to hopefully prove the accuracy of the project's hypothesis.

Bibliography

Ahlouche, Maxence (2013). *A visualisation of unclustered data compared to clustering by K-Means and K-Means++ respectively*. URL: https://www.postgresql.org/message-id/CAJeaomW50Bt-RFZs73b6M4iDaqj8mFFrfjXzHsdhSJb4iJzOHQ@mail.gmail.com (visited on 12/15/2016).

Arthur, David and Sergei Vassilvitskii (2009). "Worst-Case and Smoothed Analysis of the ICP Algorithm, with an Application to the k-Means Method". In: *SIAM Journal on Computing* 39.2, pp. 766–782. DOI: 10.1137/070683921.

Brewka, Gerhard (1991). *Nonmonotonic Reasoning: Logical Foundations of Common Sense*. New York, NY, USA: Cambridge University Press. ISBN: 0-521-38394-3.

Dublin, Trinity College (2016). *DRIS Trinity College Library Dublin*. URL: http://digitalcollections.tcd.ie/home/ (visited on 11/16/2016).

Fischer, Gerhard (2001). "User Modeling in Human–Computer Interaction". In: *User Modeling and User-Adapted Interaction* 11.1, pp. 65–86. ISSN: 1573-1391. DOI: 10.1023/A:1011145532042. URL: http://dx.doi.org/10.1023/A:1011145532042.

Halkidi, Maria, Yannis Batistakis, and Michalis Vazirgiannis (2001). "On Clustering Validation Techniques". In: *J. Intell. Inf. Syst.* 17.2-3, pp. 107–145. ISSN: 0925-9902. DOI: 10.1023/A:1012801612483. URL: http://dx.doi.org/10.1023/A:1012801612483.

Hartigan, John A (1975). *Clustering algorithms*. 1st ed. Wiley.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Kamps, Jaap and Junte Zhang (2010). "Search log analysis of user stereotypes, information seeking behavior, and contextual evaluation." In: *Proceeding of the Third Symposium on Information Interaction in Context*. DOI: 10.1145/1840784. (Visited on 11/07/2016).

Kay, Judy (2001). "User modeling for adaptation". In: *User Interfaces for All, Human Factors Series*, pp. 271–294.

Kobsa, Alfred (2001). "Generic User Modeling Systems". In: *User Modeling and User-Adapted Interaction* 11.1, pp. 49–63. ISSN: 1573-1391. DOI: 10.

1023/A:1011187500863. URL: http://dx.doi.org/10.1023/A:1011187500863.

Masthoff, Judith (2003). "Modeling the multiple people that are me". In: *International Conference on User Modeling*. Springer, pp. 258–262.

Norusis, Marija (2012). *IBM SPSS Statistics 19 statistical procedures companion*. 1st ed. Prentice Hall, pp. 375–403.

Paliouras, Georgios et al. (1999). "Exploiting Learning Techniques for the Acquisition of User Stereotypes and Communities". In: *UM99 User Modeling: Proceedings of the Seventh International Conference*. Ed. by Judy Kay. Vienna: Springer Vienna, pp. 169–178. ISBN: 978-3-7091-2490-1. DOI: 10.1007/978-3-7091-2490-1_17. URL: http://dx.doi.org/10.1007/978-3-7091-2490-1_17.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Rich, Elaine (1979). "User Modeling via Stereotypes*". In: *Cognitive Science* 3.4, pp. 329–354. DOI: 10.1207/s15516709cog0304_3.

— (1989). "Stereotypes and User Modeling". In: *User Models in Dialog Systems*. Ed. by Alfred Kobsa and Wolfgang Wahlster. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 35–51. ISBN: 978-3-642-83230-7. DOI: 10.1007/978-3-642-83230-7_2. URL: http://dx.doi.org/10.1007/978-3-642-83230-7_2.

Rousseeuw, Peter J. (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. ISSN: 0377-0427. DOI: http://dx.doi.org/10.1016/0377-0427(87)90125-7. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257.

Stockting, Bill (2004). "Time to Settle Down? EAD Encoding Principles in the Access to Archives Programme (A2A) and the Research Libraries Group's Best Practice Guidelines". In: *Journal of Archival Organization* 2.3, pp. 7–24. DOI: 10.1300/j201v02n03_02. (Visited on 11/15/2016).

Appendix A

Graphs of Number of Clusters versus Silhouette Scores

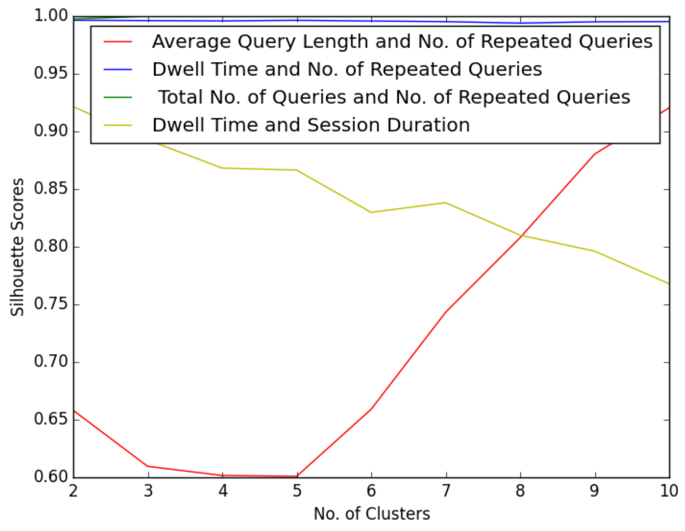

FIGURE A.1: Two Features clustered with the Unaltered Parameter

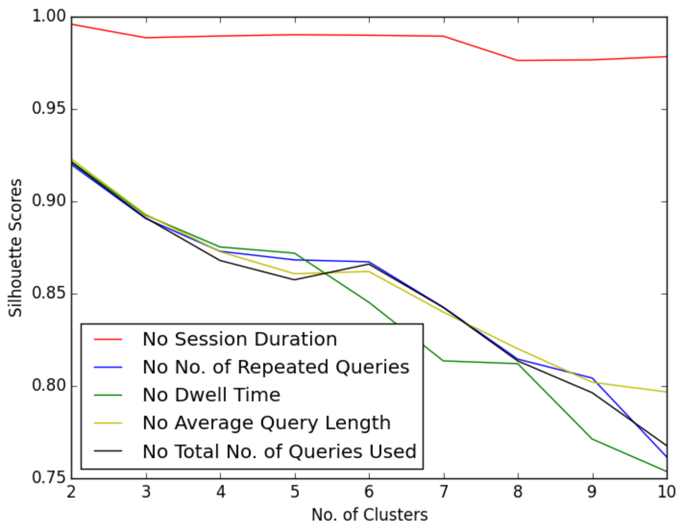

FIGURE A.2: Four Features clustered with the Unaltered Parameter

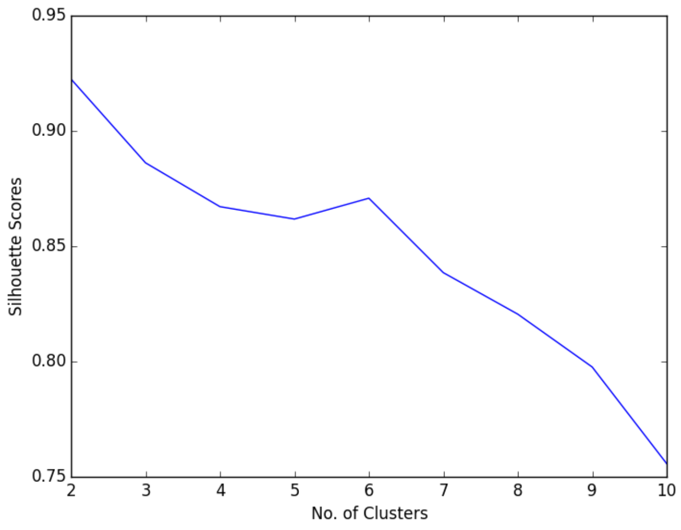

FIGURE A.3: All Five clustered with the Unaltered Parameter

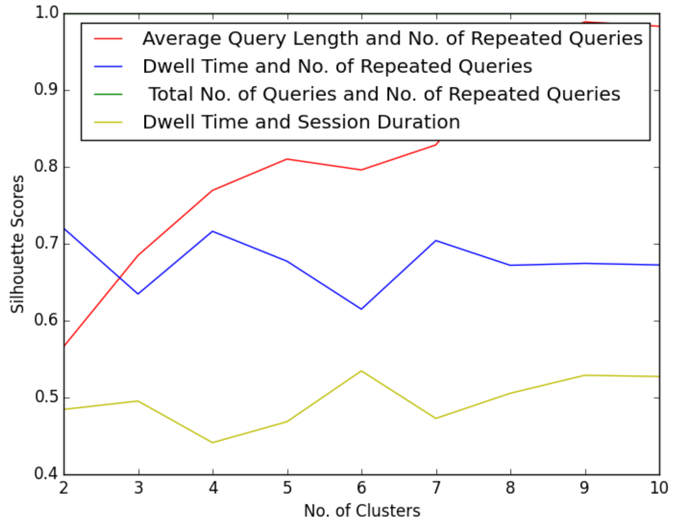

FIGURE A.4: Two Features clustered with the Filtered Parameter

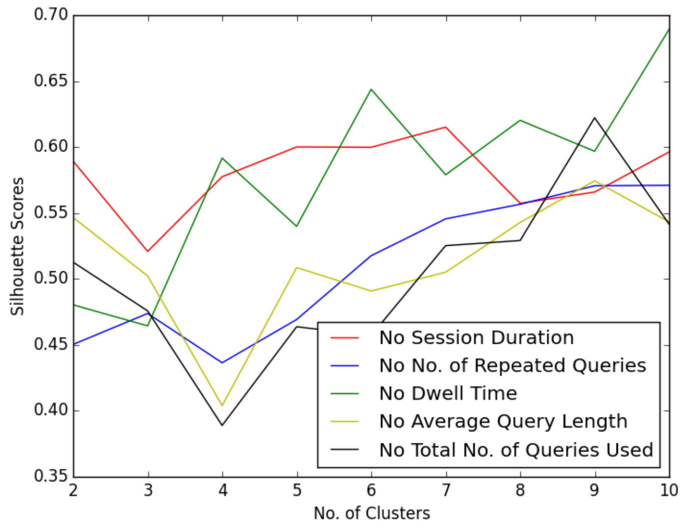

FIGURE A.5: Four Features clustered with the Filtered Parameter

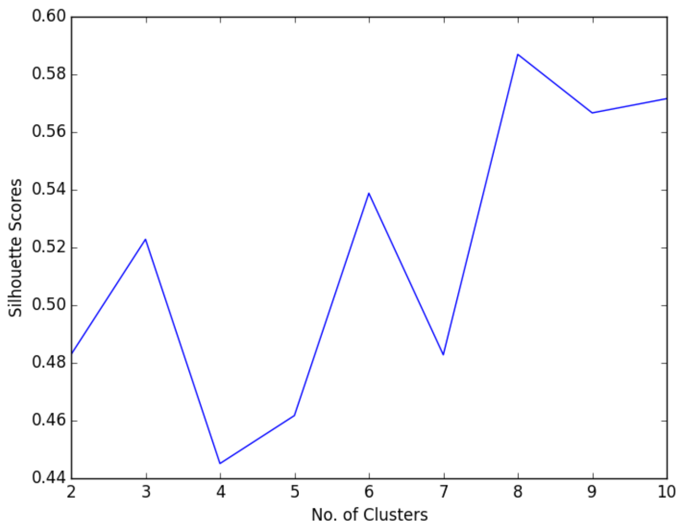

FIGURE A.6: All Five clustered with the Filtered Parameter

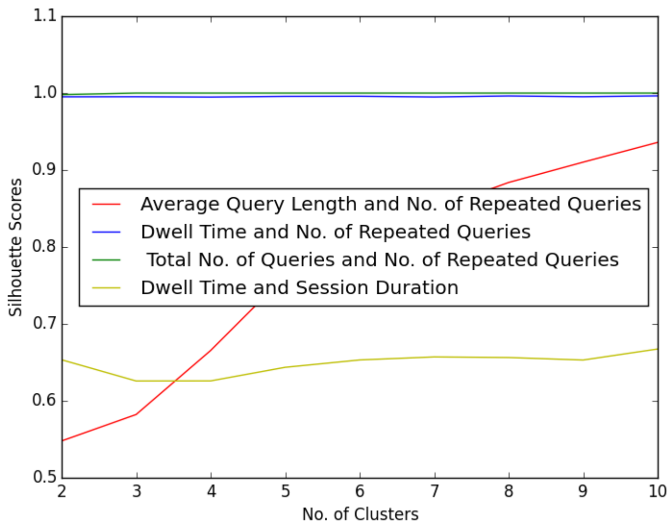

FIGURE A.7: Two Features clustered with the Logarithm Parameter

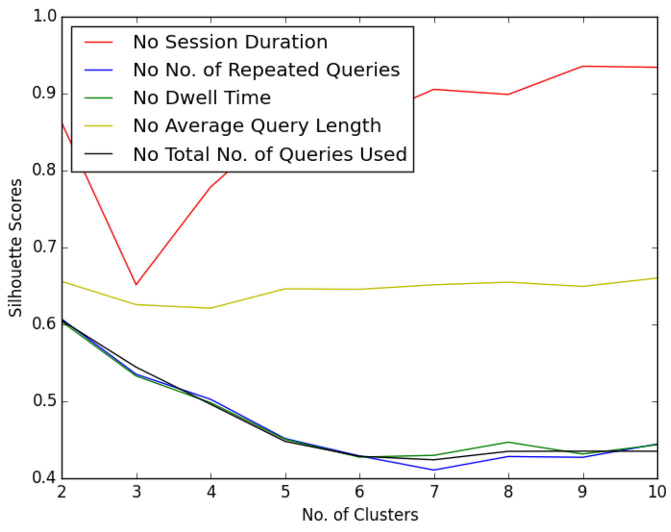

FIGURE A.8: Four Features clustered with the Logarithm Parameter

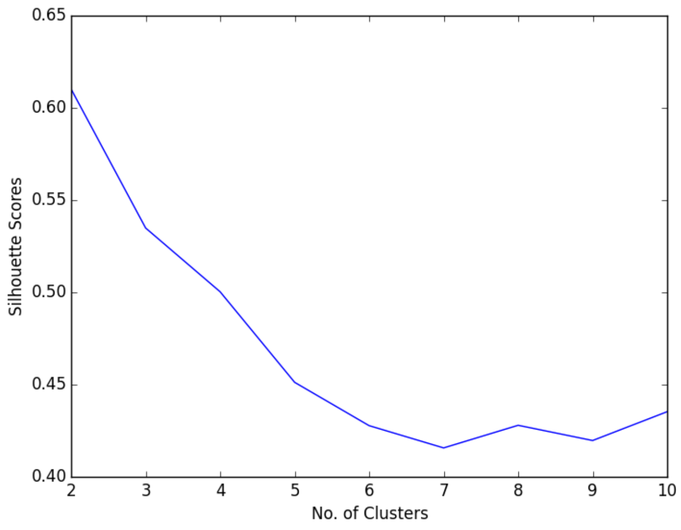

FIGURE A.9: All Five clustered with the Logarithm Parameter

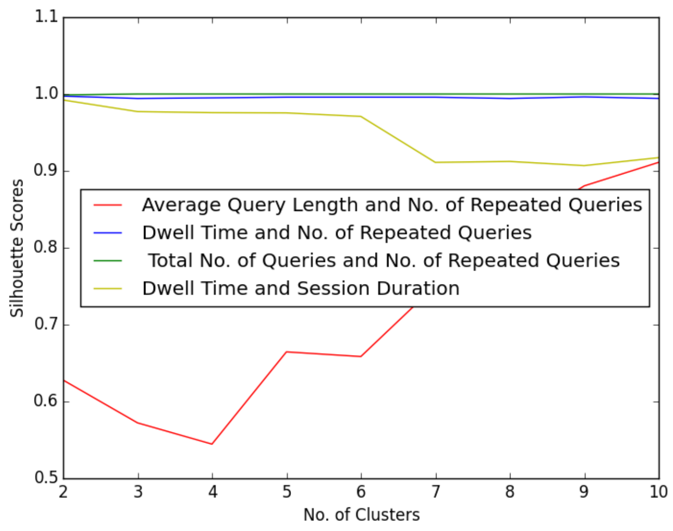

FIGURE A.10: Two Features clustered with the Normalised Parameter

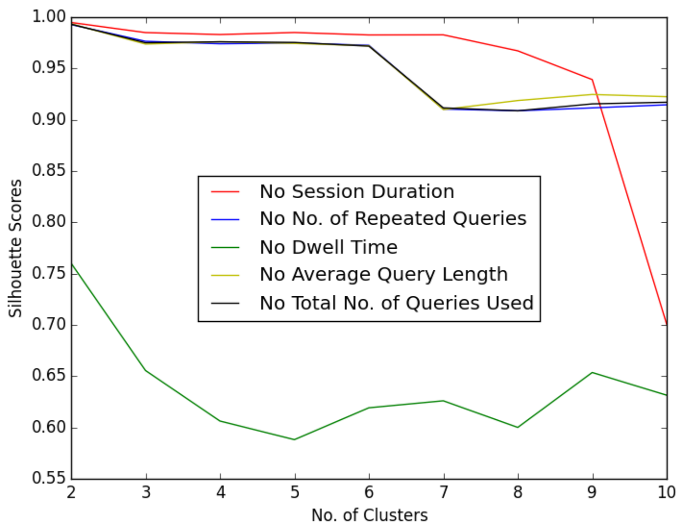

FIGURE A.11: Four Features clustered with the Normalised Parameter

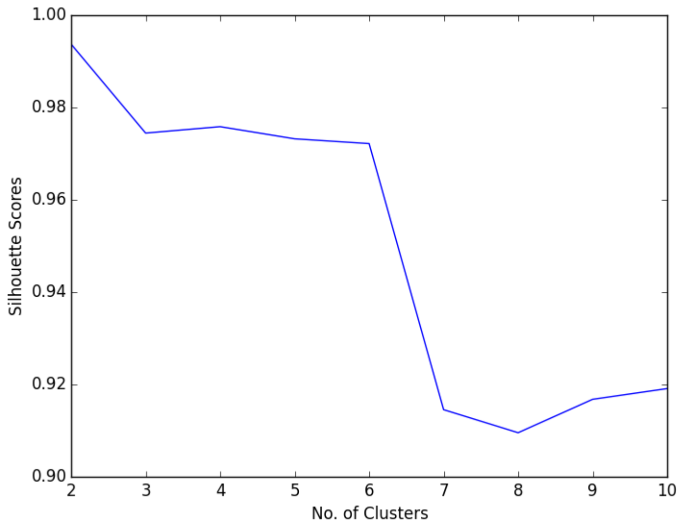

FIGURE A.12: All Five clustered with the Normalised Parameter

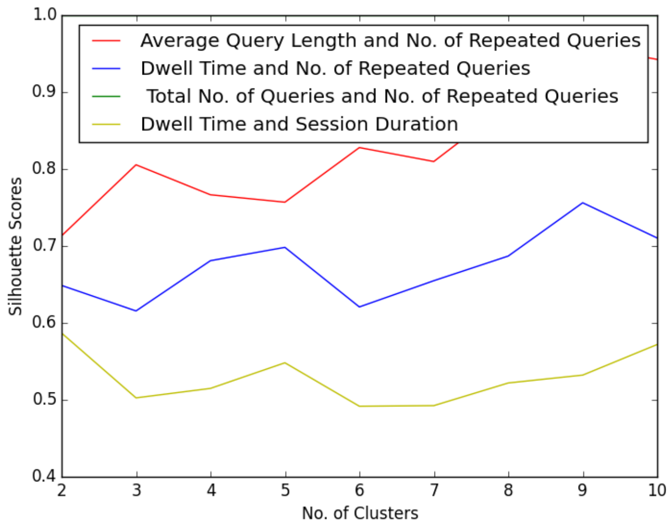

FIGURE A.13: Two Features clustered with the Normalised Filtered Parameter

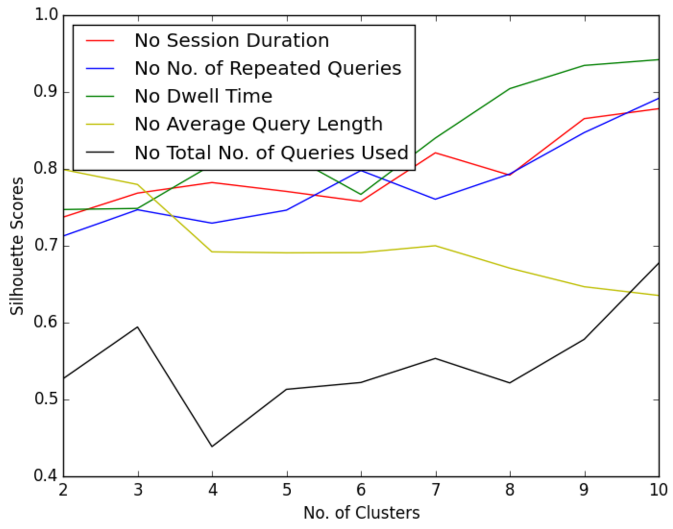

FIGURE A.14: Four Features clustered with the Normalised Filtered Parameter

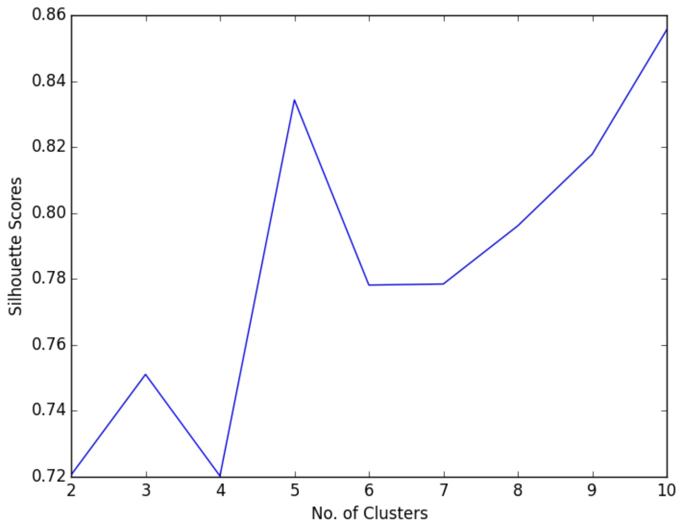

FIGURE A.15: All Five clustered with the Normalised Filtered Parameter

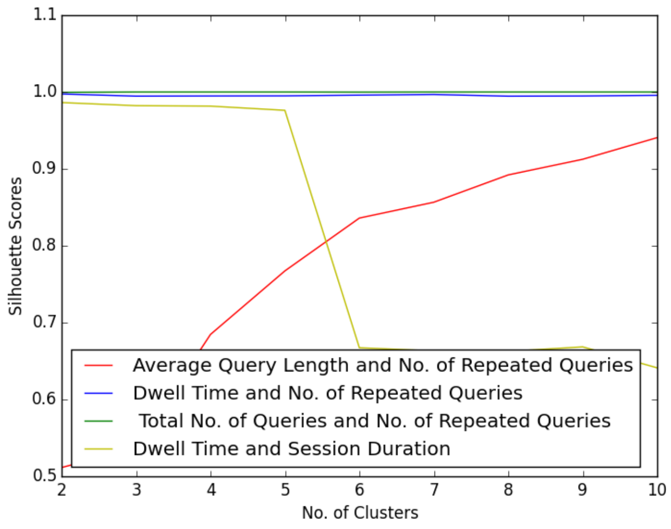

FIGURE A.16: Two Features clustered with the Normalised Logarithmed Parameter

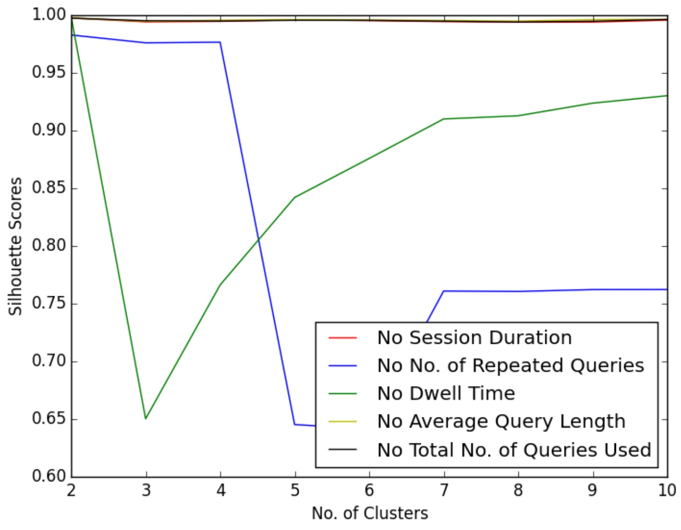

FIGURE A.17: Four Features clustered with the Normalised Logarithmed Parameter

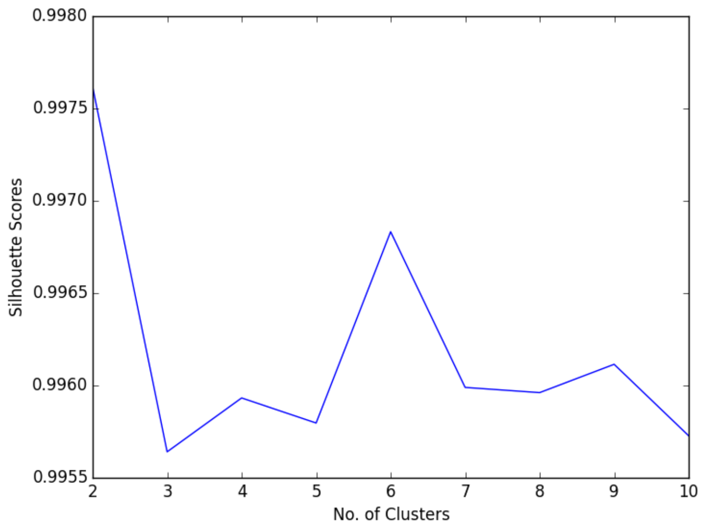

FIGURE A.18: All Five clustered with the Normalised Logarithmed Parameter