Evaluation of Test Collections Generated from Transaction Logs in Digital Humanities

A DISSERTATION PRESENTED

BY

NEIL HICKEY

TO

THE SCHOOL OF COMPUTER SCIENCE AND STATISTICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF COMPUTER SCIENCE

IN THE SUBJECT OF

KNOWLEDGE AND DATA ENGINEERING

SUPERVISED

BY

DR. SÉAMUS LAWLESS

Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

MAY 2017

Declaration

I, Neil Hickey, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signed: _____

Date: _____

Summary

The rise of digital collections being made available publicly online has led to a greater desire to give users the most relevant documents for their needs. These relevance judgments must be bench-marked to determine how well they satisfy the needs of the searcher, which involves the use of a test collection. A test collection provides a set of information needs and the documents which have been deemed to satisfy this need.

This dissertation uses two digital collections from the Cultura Project, namely the 1916 and the 1641 collections. These digital collections are a set of witness testimony documents related to key events in Irish history. The proposed experiment uses these collections as a means to evaluate an automatically generated test collection.

In this dissertation, an evaluation of whether extracted binary judgments of relevance, from transaction logs, are suitable for the automatic generation of a test collection from the digital libraries transaction logs, has been carried out. Binary judgments are simple yes or no answers to whether a document is relevant to a query or not.

To generate the test collection, a three stage experiment is proposed, which together form an end-to-end pipeline for automatic generation of test collections. The decision to split this into three parts was based on the stages of transaction log analysis in literature. The three stages take transaction logs from a source and bring them to a point where they can be analysed.

The first stage, *collection* was developed to maximise flexibility and security of the pipeline. The main functionality of this, is the extraction of the transaction logs from files using *Logstash* and the storing of this data into a scalable datastore, *Elasticsearch*. This architecture enables the experiment carried out to be applied to other digital collections in the domain. It also provides a scalable solution for large datasets.

The *preparation* stage of the pipeline, was designed to extract the relevant information needed to build a test collection from transaction logs. Built using *Python*, this application enables the extraction of relevance judgments and information needs from the datastore, and provides the final stage of the pipeline with the information it needs to perform its role.

The final element of this experiment is the *analysis* or evaluation of the generated collection. An application was built to interface with *Lucene*, which would allow the document collection to be searched over and to issue queries

against using a number of ranking functions. The *Java* based application outputs the relevance judgments for each query from both the transaction logs and the ranking functions, which facilitates the automatic generation of their performance on the collection.

The resulting performance comparison showed that each of the collections, which this experiment used to evaluate the technique, were quite different to one another. The 1641 collection resulted in statistically significant differences in the overall performance of the ranking functions, however they did not conform to the expectational ranking as set out during the design phase. The test collection built using the 1916 collection performed as expected, however the differences between ranking functions were not significant.

It was concluded that the approach of generating test collections from transaction logs proved ineffective for these sets of collections. Further work is required to provide a better set of relevancy judgments, however the issues of each collection could make it unfeasible to deem such a test collection through an automatic process.

Thesis advisor: Dr. Séamus Lawless Neil Hickey

Evaluation of Test Collections Generated from Transaction Logs in Digital Humanities

Abstract

Understanding relevance in Information Retrieval is a difficult task. Measuring the effectiveness of an information retrieval system is predicated on knowing whether a document is relevant to a search query, or not. However, relevance is subjective, personal and dynamic. Digital Humanities poses particular challenges with regards to relevance, due to smaller sized collections and the nature of these collections, amongst other factors.

This dissertation proposes and evaluates a pipeline which hopes to generate a test collection, which is used to evaluate these information retrieval systems, from transaction logs for digital libraries in the domain of Digital Humanities.

The proposed experiment is implemented using methodologies from *transaction log analysis*, and the generated test collection is evaluated through a comparison of this generated collection and expected results, as seen in literature, for two real-world digital library datasets.

This dissertation concludes that it was not possible to derive a test collection by using binary judgments of relevance from transaction logs, for the particular collections examined.

Contents

List of Figures

Acknowledgments

I would like to firstly express my gratitude to my supervisor, Dr. Séamus Lawless, for all his help and guidance throughout the course of this dissertation. I would also like to give special mention and express my sincere gratitude to both Mr Gary Munnelly and Dr Annalina Caputo, who provided me with both support and motivation throughout the course of this dissertation.

I would also like to acknowledge the support and encouragement of my family and friends throughout this year.

1 Introduction

1.1 Motivation

Web search has become an increasingly important issue in the area of digital humanities. With the rise of large collections of historical documents being made available to the public digitally [Lynch, 2002], making decisions about how to best deliver the most relevant documents to a user whilst they are searching the collection is an important issue. These digital collections provide a wealth of information which can be diverse in nature and thus raise challenges around search and relevancy. Search in digital humanities is different from web search, given for example the smaller size of the collection, the nature of documents, which can be written in an archaic English or being made principally from images, metadata and other forms.

An information retrieval (IR) system is a system which provides the user with

documents which are most *relevant* to their information needs. An information need is the topic about which the user desires to know more, and is different from a *query*, which is what the user conveys to the computer in an attempt to communicate the information need. A document is deemed to be relevant if the user perceives the document as containing information of value with respect to their personal information need, not because it just happens to contain all the words in the query.

Determining which documents are most relevant to a user is a hard problem, and studied greatly in literature, [Saracevic, 1976]. How can relevance be *measured* in IR systems? Relevance can be approximated by computing the similarity of a search query to a document. The main problem in relevancy is how to quantify this score. This dissertation will refer the term relevance to the "system / algorithmic relevance" that is the indication of the similarity to the query, whilst relevance as mentioned previously is defined as the "topical relevance".

To measure the effectiveness of ad-hoc IR systems which generate these scores, a test collection is built. A test collection provides a standardised way of comparing the effectiveness of retrieval systems. Test collections consist of a set of topics or information need descriptions, a set of information objects to be searched, and relevance judgments indicating which objects are relevant for which topics [Scholer et al., 2016].

Building a test collection is a long and expensive process. [Ritchie et al., 2006]. There are a number of different methods for generating test collections, such as that used by the Text REtrieval Conference (TREC), one of the biggest conferences that focus on the evaluation of IR systems, in which humans devise queries specifically for a given set of documents and make relevance judgments on pooled retrieved documents from that set [Voorhees and Harman, 2005]. These devised queries are created manually, including both a description of the information need, and a narrative of what classifies a document as being relevant to that need. This process is then accompanied by a group of human assessors who are given a set of documents for each information need and provide a

relevance judgment for each document. This process of generating such a test collection is unfeasible for smaller (less well funded) archives in the digital humanities area, due to the resources required, both in terms of cost and time. This dissertation investigates the suitability of transaction logs to provide this dedicated test collection where there is none available or it is impractical to build one, for the domain of digital humanities.

Transaction Logs are a record of pages being requested by a user of the system. For example, when a user enters a query into the search box of a collection, this query is placed into the request made to the server for documents to return. This provides real world interactions with a system and contains key information about what users are looking for, which pages they visit and which items they click on, allowing for an unobtrusive method of studying these queries and the results which appear to satisfy them. These clicks which appear in the transaction logs could be an indication of relevance. Therefore, if a query is an expression of an information need and a click could indicate relevance, is it possible to generate a rough evaluation collection for IR systems using these transaction logs? This dissertation investigates the building of a test collection to evaluate IR systems using transaction logs, by segregating and filtering this wealth of interactions to generate relevancy judgments which can be used as an alternative to a human generated, dedicated test collection, where resources make this unfeasible.

1.2 Research Question

Evaluation is highly important for designing, developing and maintaining effective information retrieval or search systems as it allows the measurement of how successfully an information retrieval system meets its goal of helping users fulfil their information needs. IR Systems typically use a user generated test collection to measure the effectiveness of the system. An approach as used by TREC in some evaluations, is the notion of giving a binary classification as either relevant or nonrelevant for an information need with respect to a document in the test collection. This decision is referred to as the *gold standard* or *ground truth*

judgment of relevance, [Manning et al., 2008]. However for small digital libraries in the domain of digital humanities, this manual process is *expensive* or *impractical*. I will investigate the effectiveness of using transaction logs to generate this test collection. These transaction logs are automatically generated, thus having a large reduced cost over manual generation, although there are still costs involved such as time to grow the logs and the costs involved with mining this data. This dissertation aims to evaluate this proposal by addressing the following question:

Can transaction logs provide a suitable *alternative test collection* for evaluating Information Retrieval Systems in Digital Humanities?

Challenges:
There are a number challenges which arise by addressing this question such as the suitability of binary relevance judgments and segregation of user interactions with the system using inference based on a limited context. Verifying the effectiveness of this generated test collection is key to determining the validity of transaction logs as a means to generate test collections, and finally the ethical implications involved with analysing real user interactions with a system.

1.3 Use Case - The Cultura Project

This dissertation initially began using the Trinity College Dublin Digital Collections library as a use case, however after much analysis, was deemed unsuitable. Chapter 6 discusses the reasons for this and the change to using the Cultura Project, which became the main focus of the experiment.

The Cultura Project is an EU project, which aims to "pioneer the development of next generation adaptive systems which will provide new forms of multi-dimensional adaptivity."

The following are two collections of historical documents which are made available by the project, both are available online and have transactional logs on

their servers. The transactional logs of both of these collections have been kindly provided in aid of fulfilment of this dissertation.

1916 Collection
The Bureau of Military History documents the movement for Independence from November 1913 to July 1921. The bureau contains witness statements, sets of contemporary documents, photographs, voice recordings, and a collection of press cuttings. These were largely contributed by participants or witnesses to the events. This collection has 119 documents.

1641 Collection
The 1641 Depositions are witness testimonies, from all social backgrounds, concerning their experiences of the 1641 Irish rebellion. The testimonies document the loss of goods, military activity, the alleged crimes, and the causes and events surrounding the 1641 rebellion. This collection contains 8129 documents.

1.4 Goal of Dissertation

The goal of this dissertation is to determine whether user clicks can be interpreted as absolute judgments of relevance, using transaction logs to extract these judgments. The purpose of test collections is for evaluating how well a *ranking function* performs on an IR system. These ranking functions are used to create a score which attempts to represent the *relevance* of documents to the query issued by a searcher. Test collections are widely available for web search, however this is not the scope of this experiment, which aims to aid small digital libraries in the evaluation of their IR systems using a more cost-effective method of generating the test collection.

The first step to achieving this is the extraction of information from transaction log data, how this is designed and the methodologies used are discussed in the following chapter. Once this test collection is available, a key component is the

evaluation of this generated collection, to determine it's effectiveness. This evaluation is a comparison of the performance of a number of ranking functions on the generated test collection against what has been achieved in literature. This is not an evaluation of the ranking functions themselves, rather a comparison of their performance on the test collection. The assumption is that there is an expected result of how well these functions perform in relation to one another, and thus if this assumption holds, a reasonable conclusion is that the test collection reflects the information needs and fulfilment of those needs well.

1.5 Overview of Dissertation

Up to this point, the purpose of test collections in Information Retrieval has been outlined, alongside the motivation for this dissertation, the research question to be answered and the challenges involved in answering that question. The following chapters outline the design and execution of this experiment.

Chapter 2 discusses relevant historical work in the area of log analysis. This is followed by a discussion of the state-of-the-art work being carried out, and the performance of similar solutions.

Chapter 3 outlines the design of the proposed experiment, including the architectural decisions and scalability factors which are accounted for when performing analysis of large datasets. The methodology of building an automated test collection is discussed and the evaluation stage design.

Chapter 4 describes the details of the technical aspects of the framework. The topics discussed in this chapter span from the ingestion of transaction logs, the development of the application to aggregate and filter the data to generate relevancy judgments, and finally the Lucene-based application development for evaluation of the generated test collection.

Chapter 5 presents the results of evaluation, as well as the performance metrics used to determine the effectiveness of a generated collection. This chapter critically analyses performance of the various ranking functions in relation to the test collection, and by extension the feasibility of generating such a test collection

from transaction logs.

Chapter 6 contains some final thoughts on the dissertation as a whole. This chapter discusses the achievements of the dissertation, as well as the limitations and obstacles encountered. Finally, future work to be explored is also discussed.

"In one word he told me secret of success in mathematics: Plagiarize! ... only be sure always to call it please, 'research'."

- Tom Lehrer, Lobachevsky

2

Literature Review

At the beginning of this dissertation, both design and methodology choices had to be made to construct an experiment which could answer the question that this dissertation poses. The technologies and methodologies chosen for use within this dissertation were made through contextual research into a number of domains, such as transaction log analysis, search query segmentation in logs, evaluation of test collections and data privacy. This chapter aims to discuss the state-of-the-art approaches to problems similar to those discussed within this dissertation, alongside a discussion of their methodology and results, and how these impact on this dissertation.

2.1 Transactional Log Analysis

The area of transaction log analysis for online public access catalogues has been widely adopted in literature. Web search engine companies use transaction logs to research trends and effects of system changes. Transaction logs are an unobtrusive way of gathering search and interaction data from system users [Jansen, 2006]. The process of conducting an examination of this collected data is referred to as transaction log analysis (TLA). In the context of my dissertation, an analysis of the shortcomings and strengths of using TLA as a methodology to build a test collection, for evaluating IR on Cultura Heritage datasets, are highlighted and discussed.

"Using TLA as a methodology, one examines the characteristics of searching episodes in order to isolate trends and identify typical interactions between searchers and the system. Interaction has several meanings in information searching, addressing a variety of transactions including query submission, query modification, results list viewing, and use of information objects (e.g., Web page, pdf file, video)" [Jansen, 2006].

Figure 2.1.1: Transaction Log Analysis Pipeline

TLA involves the following three major stages, which are shown in figure 2.1.1:

1. **Collection:** the process of collecting the interaction data for a given period in a transaction log.
2. **Preparation:** the process of cleaning and preparing the transaction log data for analysis.

3. **Analysis:** the process of analysing the prepared data.

Strengths of TLA

TLA provides a method of collecting data from a vast number of users. Given the scale of the Web, the approach appears to be the most unobtrusive method of collecting user-system interaction from searchers.

The costs of collecting data is low. The interactions between users and the system are logged into files, where the cost is storage and maintenance. There are minimal costs involved in collection / generation of this data, as they are real interactions which are simply being recorded.

It is important to emphasise that the collection of these user-system interactions is unobtrusive. Interactions between searchers and the system are not artificial or for the purpose of experimentation. Transaction logs represent the unaltered behaviour of searchers, where the searchers are issuing genuine queries and judging relevancy in the absence of a controlled environment. This can reduce or eliminate bias which could influence the outcome of experiments.

This is an important consideration for this dissertation, the judgments can be different or may in fact be more appropriate when generating a test collection in comparison to a manually generated set of judgments. Manual relevance judgments are given by a limited number of participants, therefore the potential for gaining different insights into relevance judgments as deemed by a large quantity of searchers is quite large and could be the basis of improving judgments deemed through TLA.

Weaknesses of TLA

Almost from its first use, researchers have critiqued TLA as a research methodology [Belkin et al., 1995]. It is reported that transaction logs do not record the users' *perceptions* of the search, cannot measure the underlying the *information need* of the searchers, and cannot gauge the searchers' *satisfaction* with search results, [Borgman, 1996]. Kurth notes that transaction logs can only deal with the actions that the user takes, not their perceptions, emotions, or

background skills [Kurth, 1993].

Kurth further identifies three methodological issues with TLA: *execution*, *conception*, and *communication*. TLA can be difficult to execute due to collection, storage, and analysis issues associated with the complexity and quantity of the data set. With complex data sets, it is difficult to develop a methodology for analysing the dependent variables. Communication problems occur when researchers do not define terms and metrics in sufficient detail to allow other researchers to interpret and verify their results.

However, these are issues with many, if not all, empirical methodologies. Although the points made by Kurth are still valid today, advances in transaction logging software, transaction log format standards, and improved data analysis software and methods have addressed many of these shortcomings. This dissertation aims to investigate the use of transactional logs as a means to generate test collections. As transaction logs only provide a limited context as described by Kurth, techniques inferring information from logs are considered to be a mechanism for comparing system performance and not designed to give a 'perfect relevance' judgment. This is important in the context of this dissertation, the aim is to build a test collection from limited context and to evaluate how well it performs. The limitations and expensive process of generating such a collection manually are significant enough to warrant investigation of such an automated process through the usage of TLA.

2.2 Extracting Information

There are many ways of extracting information on clicks or queries from a transaction log to make a topic set and associated relevance judgments. A simple way would be to treat every query typed by a user as a topic, and every result that the user clicked on as a positive relevance judgment. However, such an approach may not lead to a good test set [Arampatzis et al., 2007]. Previous research on user click behaviour has shown that clicks on search engine results do not directly correspond to explicit, absolute relevance judgments, but can be considered as

relative relevance judgments [Susan Dumais, 2003]. For example, if a user skips result *A* and clicks on result *B*, then the user gives preference to result *B*, indicating that result *B* should have a higher relevance judgment to result *A*. This was an important consideration for this dissertation, can user clicks be interpreted as an explicit indication of relevance and if so, can they be used to generate a test collection? The findings of Dumais are an important factor for this dissertation. The methodology for building judgments from user interactions with the system is key to creating a good test collection, therefore careful consideration of the paper and its findings were accounted for in the design stage of this dissertation. This dissertation aims to investigate the claim that explicit judgments may lead to a poor reflection of relevance judgments and to show the viability of building relevance judgments from transaction logs and as such the experiment uses a simple click model, however as noted above careful consideration of the effects of such modelling is taken and will be the focus of future work.

Defining Relevance - Sessions

Determining which documents are relevant to a query defines the way that documents, which appear to satisfy the information needs of a search, are deemed to be relevant. Transaction logs maintain no notion of a user starting and ending their information needs. A method for attempting to understand when this occurs is known as defining user *sessions*, which is an important consideration when generating test collections from transaction logs. A session is designed to include all documents deemed relevant to a query or information need as required by a system user. Previous research into the area of using transaction logs for test collections by Arampatzis [Arampatzis et al., 2007], provides a number of definitions of relevance based on the contents of transaction logs:

- **Raw queries:** the bag of queries. Every query, along with it's corresponding clicked results from one session.
- **Unique union:** the set of queries. All results clicked by all users for the set

of queries are considered relevant.

- **Unique intersection:** the set of queries. Unique union set where all users have clicked on a result.

Arampatzis's paper showed that ranking based on the Raw Topic set deviates slightly from ranking based on the Union and Intersection topic sets. The Union and Intersection topic sets result in exactly the same ranking. This dissertation looks into the Unique union set of queries and relevance judgments. Future work for this dissertation would aim to incorporate these other definitions and perhaps a weighted mixture of relevance judgments, to deem which performed most consistently.

2.2.1 User Goals and Session Boundaries

Most analysis of web search relevance and performance takes a single query as the unit of search engine interaction. However a number of studies have shown that the usage of such units of interaction may be a poor reflection of the users thoughts and expressions of interest. When studies attempt to group queries together by task or session, a timeout is typically used to identify the boundary. However, users query search engines in order to accomplish tasks at a variety of granularities, issuing multiple queries as they attempt to accomplish tasks [Jones and Klinkner, 2008]. Jones discusses three methods of defining user goals, as outlined:

1. A search session is all user activity within a fixed time window.
2. A search goal is an atomic information need, resulting in one or more queries.
3. A search mission is a related set of information needs, resulting in one or more goals.

In order to ascertain the viability of the general approach to generating test collections through the use of transaction logs, this dissertation focuses on the first method, as defined by Jones. However, the discussion of defining a search mission is important to the future work and research of this experiment.

The other approaches as described by Jones include the notions of *goals* and *missions*. A goal is a group of related queries to accomplish a single discrete task, whereas a mission is an extended information need such as finding a number of restaurants, for a searcher interested in places to eat locally. Figure 2.2.1 below illustrates an example of a hierarchy which contains goals and missions.

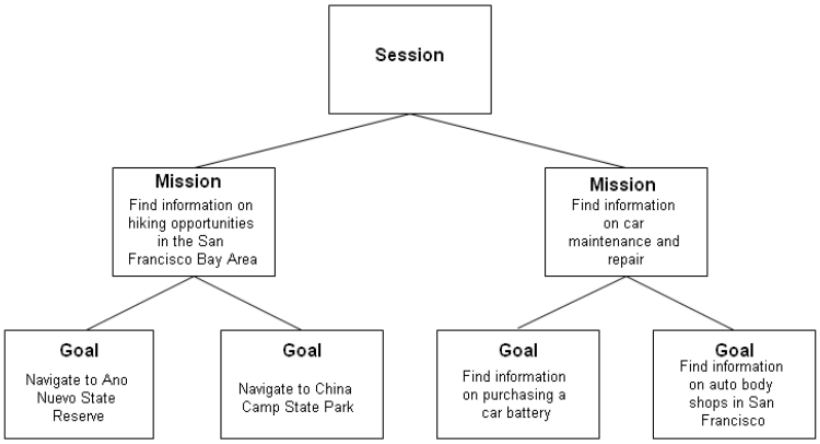

Figure 2.2.1: Sample hierarchy of user missions and goals [Jones and Klinkner, 2008]

Jones discusses the use of hierarchical tasks, and the usage of boundary timeouts for sessions and the weaknesses of using a boundary time regardless of length. As shown in figure 2.2.2 Jones concludes that boundary timeouts are of limited utility in identifying task boundaries, achieving a maximum precision of only 70%, and that the choice of timeout is relatively arbitrary.

Time threshold	Goals	Missions
	Boundary	Boundary
Baseline	54.2%	70.9%
5 minute	71.2%	75.6%
30 minute	66.5%	78.6%
60 minute	64.2%	77.6%
120 minute	62.0%	76.1%
Trained time	71.2%	78.6%

Figure 2.2.2: Mission boundaries

Jones concludes that through the usage of goals and milestones, an improvement on timeout segmentation was achieved, bringing the accuracy up to 92% for identifying fine-grained task boundaries, and 89-97% for identifying pairs of queries from the same task when tasks are interleaved hierarchically. This improvement over timeout based sessions is quite important in the context of this dissertation, with implications for the success of a generated test collection from transaction logs.

2.3 Summary

In this chapter various state-of-the-art technologies were discussed alongside the strengths and weaknesses of the approach as illustrated in literature. Of those discussed, this project aims to build a TLA based approach to generating test collections. The modelling approach to session generation is proposed to be based on timeouts as described by Jones. This is a relatively simple approach to generating relevance judgments.

Despite the limitations of TLA as a methodology, for the domain of digital humanities and given the unique nature of the document collections, the usage of TLA to evaluate IR systems was justified. Analysing the transaction logs and using them to evaluate IR systems in digital humanities will yield slightly better quantitative results than existing methods in the area. The flexibility of the proposed framework design discussed in the following chapter as a means to generate this test collection, gives a foundation to extend the findings of this

dissertation using approaches found in state-of-the-art studies.

"Design is not just what it looks like and feels like. Design is how it works."

- Steve Jobs

3 Design

3.1 Introduction

The main goal of this dissertation is to determine whether user clicks can be interpreted as absolute judgments of relevance, using transaction logs to extract these judgments. The design and methodology choices involved in this experiment focus on the capturing of data available in digital libraries transaction logs, builds a set of queries and relevance judgments from this data and finally evaluates the effectiveness of this generated test collection. The main design aspects of this dissertation are around the methodology of generating a test collection, the choice of evaluation techniques and finally the technical choices and architecture of the developed system.

3.2 Test Collection Generation

To evaluate a test collection generated via transaction logs, the three components of a test collection must be available. Outlined below are the three key elements to a test collection, and the methodology of this dissertation for generating them via transaction logs.

3.2.1 Document Collection

A document collection is the set of documents which users will query against. In the case of the Cultura Project, these are historical documents relating to important events in Ireland and witness testimonies of these events.

The 1641 collection contains 8129 documents, which are transcribed historical witness statements from the 1641 Irish rebellion. These documents have been transcribed from paper and are written in an archaic form of English, which is different from the form of English used now. The 1916 collection is composed of 119 documents, which are made up of witness statements from the 1916 rising and surrounding events.

3.2.2 Information Needs (Queries)

To evaluate the performance of an IR system, a set of *queries* are required which represent information needs of users for the system. A query is a set of one or more *terms* (keywords). Through transaction logs, the search queries issued by users can be discovered. If a transaction log line contains a query, it is added to the set of all queries issued against the system. Figure 3.2.1 below shows an excerpt from the transaction logs, showing an example user search query and the subsequent clicks which they performed to fulfil their information need. In this example, the user issued the query "Parnell Street", and subsequently clicked on two documents - WS0242 and WS1709, within the timeout boundary.

The design of the overall process by which these queries would be extracted from the transaction logs is discussed in depth in section 3.4.

```
188.143.232.11 - - [25/Dec/2016:19:05:14 +0000] "GET /1916/
q=search/1916results&searchQuery=%22Parnell%20Street%22 HTTP/1.1" 200 26091 "-" "Mozill
a/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
188.143.232.11 - - [25/Dec/2016:19:05:16 +0000] "GET /1916/ HTTP/1.1" 200 14407 "-"
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
188.143.232.11 - - [25/Dec/2016:19:05:19 +0000] "GET /1916/?q=artefact/WS0242 HTTP/1.1"
200 126030 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
188.143.232.11 - - [25/Dec/2016:19:05:21 +0000] "GET /1916/?q=artefact/WS1709 HTTP/1.1"
200 131161 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
```

Figure 3.2.1: Example query and clicks from transaction log

3.2.3 Relevance Judgments

To evaluate the performance of an IR system, a set of *relevance judgments* are also required. For this dissertation, these judgments are simple judgments of whether a document is relevant or not to a particular query. Building the relevance judgments is the most important aspect of a test collection and requires most effort for manually generated collections.

The definition of relevance is a key factor in generating judgments. How does this experiment define relevance? The topic of sessions has been introduced in Chapter 2, with different methodologies for defining a session. The following section details how this experiment deems a click to be relevant to a query as issued by a user.

Session Design

As described previously, a session is a period of time for which a user is pursuing a particular information need. This dissertation defines a session as being all clicked documents which occur by a user between the issuing of one query to the next, with a maximum timeout limit of one hour. As discussed in the Chapter 2, this session boundary may be relatively arbitrary, however for the purposes of showing the viability of the approach, a simple boundary-based approach was taken.

The approach taken in this experiment is that any query issued creates a new session. This session is concluded when the same user issues another query or

the boundary of one hour has been reached. Each session can therefore only have one unit of information need (query). Any clicked documents after the issuing of a query are added to the session, and deemed relevant to satisfying that query. Figure 3.2.2 illustrates a flow diagram of the designed session generation process.

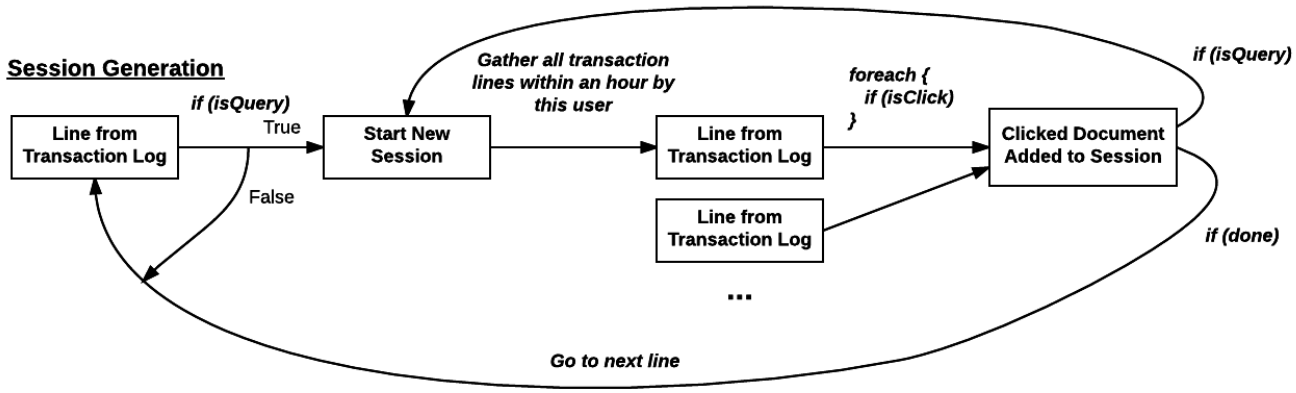

Figure 3.2.2: Session Generation Design

Figure 3.2.3 below shows a resulting session from the snippet shown in figure 3.2.1 and it's associated relevance judgments, as generated by the automated system, where the maximum boundary time between document 'WS0242' and 'WS1709' is one hour.

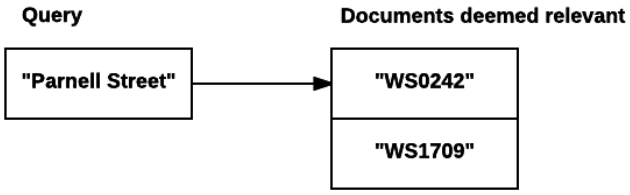

Figure 3.2.3: Sample Session

3.2.4 Evaluation of the Test Collection

The key question being answered by this question is around the evaluation of a generated test collection through transaction logs. What is the purpose of evaluation in this context? Evaluation of a generated test collection involves the *comparison* of a number of ranking functions performance. The evaluation stage of this experiment is not designed to see how well each ranking function performs, but rather how they do in relation to one another.

The three chosen ranking functions, namely the Vector-Space Model, BM25 and Jelinek-Mercer Smoothing, are expected to perform in this order, respectfully. BM25 has been shown to work well in literature on content of digital libraries. A study by Bennet shows that smoothing performs very well when tuned to the an empirically selected level of smoothing, [Bennett et al., 2007] and is expected to perform best on the collections.

3.3 Development Stack Design

The overall goal of this experiment is to generate a test collection from transaction logs, and evaluate it's effectiveness. As outlined in the previous section, the system must automatically generate a test collection, therefore the design of this system reflects processes involved in generating each component of a test collection. The overall, high-level design of the system is shown in Figure 3.3.1.

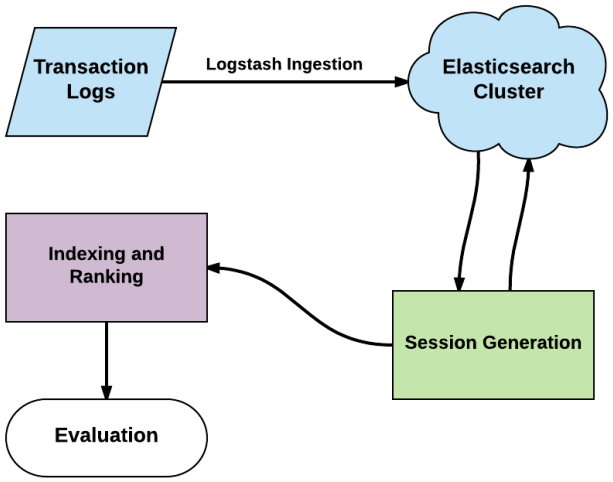

Figure 3.3.1: Overall System Design

3.4 Data Extraction & Ingestion

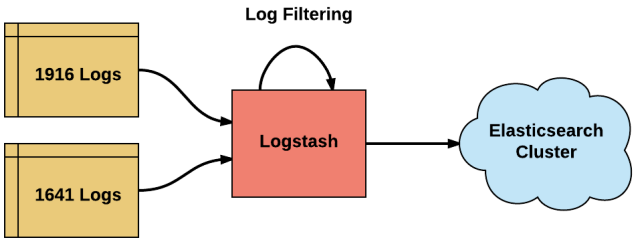

Figure 3.4.1: Logs Extraction

3.4.1 Overview

The first step to undertake is to source logs from a digital collection, with sufficient data to perform the dissertation. In conjunction with the ADAPT Centre of Trinity College Dublin, I was able to source two sets of transaction logs from the *Cultura Project*. The Cultura Project aims to guide, assist and empower every users interaction with Europe's cultural treasures. The Cultura Project makes available multiple collections; two such collections include the '1916

Rising Collection' and the '1641 Depositions'. By applying the approach outlined by this dissertation to two sets of collections, I can strengthen the argument that the approach taken in with dissertation indicates transference over different collections.

The Cultura Project transaction logs contain over 1.2 million transactions, with 500 thousand entries related to 1916 and 460 thousand entries related to 1641. The contents of the logs spans a total of 4 years from June 2013 to March 2017. The 1641 collection has a total of 1324 queries issued, with 498 unique queries. The 1916 collection has a total of

Before these transaction logs can be used, the information must be extracted into a filterable format. Logstash [1] is an open-source file parser which will index and separate the logs into query-able and filterable entities. This is an automated process, and means that no extra work is required to define these.

3.4.2 Predefined Terminology

Log files are a standard tool for computer systems developers and administrators. They record the transactions of the system, such as what resource was accessed and by whom. This information can record faults and help their diagnosis. It can identify security breaches and other computer misuse, auditing and can also be used as a means to gather statistical information on, for example, the frequency of access of resources.

The information stored is only available for later analysis if it is stored in a form that can be analysed. This data can be structured in many ways for analysis. For example, storing it in a relational database would force the data into a query-able format. However, it would also make it more difficult to retrieve if the computer crashed, and logging would not be available unless the database was available. A plain text format minimises dependencies on other system processes, and assists logging at all phases of computer operation, including start-up and shut-down, where such processes might be unavailable.

[1] `https://www.elastic.co/products/logstash`

For the Cultura Project collections, logs are in an Apache server format, known as 'The Common Log Format', which is a standardised text file format used by web servers when generating server log files. Because the format is standardised, the files can be readily analysed by a variety of web analysis programs. The following are the main components of the Apache format:

- **Client IP:** IP address of user accessing the resource.
- **Date-Time:** Date and time of access.
- **Resource:** The webpage / document accessed.
- **Status:** Status of request, standard HTTP response code.
- **Referral Uri:** If applicable, the uri of the resource which brought the user to this page.
- **Device:** Browser / Computer type used to access the collection.

An example of the format is given below:

```
1.1.1.1 - - [01/Jan/2016:01:01:01 +0100]
"GET / HTTP/1.1" 200 8024 "http://lib.xyz.com/?id=abc"
"Mozilla/5.0 (iPad; CPU OS 9_3_2 like Mac OS X)
AppleWebKit/601.1 (KHTML,like Gecko) Version/9.0
Mobile/13F69 Safari/601.1"
```

Listing 1: Example Apache Formatted Log Excerpt

3.4.3 Extraction Methodology and Design

To create the pipeline for extraction and filtering of the transaction logs into a format which is understood by the data-store, a number of pre-processing steps must be taken, as outlined as follows:

1. The Logs are sourced from a collection.
2. Each line of the logs must be filtered, so each component of the transaction is separated.
3. Each extracted entity is stored in the data-store, for further analysis.

The pre-processing stage of the pipeline is automated, requiring no user input. To generate a test collection, I wanted to introduce the minimal effort for a current system to integrate the approach taken by this dissertation. As the transaction logs are in the standardised format, simple filtering can be applied to extract the information, again requiring minimal configuration. To be a real alternative to manually generated test collections, the amount of effort required to generate relevancy judgments must be less than that of a manually created collection. This was an important factor in the design choice, with automation and minimal configuration a key aspect of the decisions made.

Each line of the transaction logs is translated into a format understood by the data-store. As shown in listing 2, each line of the transaction log is translated into a document structure, made up of many fields. Each field corresponds to a part of the transaction line or meta information about it. I made the decision to deconstruct the transaction logs into the following format as it allows for filtering and extraction of detailed information without manual transformation at a stage further along the pipeline.

An important field when extracting information is the timestamp field. This identifies the time at which a resource was accessed, as it is important when generating sessions. This timestamp includes a time zone offset which accounts for the time zone where the server resides. To simplify querying and filtering of transaction logs into sessions, a time zone should be chosen which will standardise the transaction logs timestamps. This is an important design decision for flexibility of the pipeline, and for transferability of the approach taken in this dissertation to other digital libraries.

```
"_source" : {
    "message" : "1.2.3.4 - - [17/Jul/2013:15:19:57 +0000] \"POST
    ↪  /1641/?q=deposition/809001r001 HTTP/1.1\" 302 778
    ↪  \"http://cultura-project.eu/1641/?q=deposition/809001r001\"
    ↪  \"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
    ↪  (KHTML, like Gecko) Chrome/28.0.1500.72 Safari/537.36\"",
    "@version" : "1",
    "@timestamp" : "2013-07-17T15:19:57.000Z",
    "path" : "/logs",
    "clientip" : "1.2.3.4",
    ...
    "timestamp" : "17/Jul/2013:15:19:57 +0000",
    "verb" : "POST",
    "request" : "/1641/?q=deposition/809001r001",
    "httpversion" : "1.1",
    "response" : 302,
    "bytes" : 778,
    "referrer" :
    ↪  "http://cultura-project.eu/1641/?q=deposition/809001r001",
    "agent" : "\"Mozilla/5.0 (Windows NT 6.1; WOW64)
    ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.72
    ↪  Safari/537.36\""
}
```

Listing 2: Sample Extracted Document

As shown in Listing 2, each transaction appearing in the log is filtered and split into named fields. A brief explanation of fields I am most interested in is detailed below:

- **@timestamp:** This timestamp is a GMT representation of the time this transaction occured. This converts the timestamp to GMT if it is in another time zone.
- **clientip:** The client IP is assumed to be the user who accessed the

resource, I assume that all requests from an ip are from the same person, however this may not always be the case and is a limitation of TLA.

- **request:** The request field contains information about the accessed resource, i.e the ID of the resource in the system.

Why Elasticsearch and Logstash?
For development and deployment purposes, a distributed data-store was chosen. This provides the ability to scale the TLA process to large volumes of data. Elasticsearch [2] is a document storage system which exposes it's operations and extraction via a HTTP Application Programming Interface. This provides an implementation agnostic approach to data extraction, as discussed in section 3.5.

3.4.4 Design Ethics

Usage of real-world system interactions, including identifying information raised a number of privacy issues which needed to be addressed:

1. Identification of Users
   Interactions between users and the digital library are recorded, and as such some identifying characteristics are recorded i.e. IP addresses. To mitigate this privacy implication, this dissertation looks at relevancy judgments made by all users of the system, and not individual results. This uses an amalgamation of user interactions to generate relevancy judgments, thus reducing scope for privacy concerns.

2. Security
   Another important consideration when designing an automated system is the security underpinning it, especially where user data is being recorded and analysed. It was important that information being used by the automated test collection generation system was stored securely and not available publicly. The technical implementation of this dissertation uses

[2] `https://www.elastic.co/products/elasticsearch`

the latest security standards with role based access to the system. Security and integrity of user based data is a highly important risk factor to consider for both reputational and regulatory reasons.

3.5 Session Generation

3.5.1 Overview

The extraction of relevant information from the transaction logs is important to generating different test collections. As outlined in section 3.5.3, there are a number of extraction techniques as defined by Arampatzis to extract relevance information from data available in the transaction logs.

3.5.2 Predefined Terminology

Outlined below is the set of terminology underpinning transactional log research, each identifying an important element to consider for generating relevance judgments.

- **User:** the client accessing the collection as identified by their ip-address.
- **Transaction:** any exchange between client (user) and server (system), corresponding to a line in the transaction log.
- **Session:** a sequence of transactions by the same user, where the maximum interval between transaction n and $n + 1$ is 1 hour.
- **Query:** the string typed by the user as it appears in the transaction log.
- **Identifier:** the identifier of the digital library item, used to retrieve the object data from the object database.

3.5.3 Extraction Methods

Arampatzis used a number of standard extraction methods, [Arampatzis et al., 2007], which were used to extract details about user sessions and what a user found to be relevant whilst searching:

- **Raw queries:** the bag of queries. Every query, along with it's corresponding clicked results from one session.
- **Unique union:** the set of queries. The same query addressed by different users is aggregated into a single one, i.e. the relevance judgments are accumulated across all users searching with the same query.
- **Unique intersection:** the set of queries. The same query addressed by all users is aggregated into a single one, i.e. a result is relevant only if all users who typed the query, clicked on that result.

The decision was made to generate sessions using the Unique Union set as described above, with the implication that future work would involve extending to other methodologies.

3.5.4 Application Design

An important aspect of generating a test collection are user clicks. Clicked pages within a user session are deemed as relevant, however I make no particular claims on the interpretation of clicks. I assume that the searcher found these pages interesting enough to look more closely at, and that a more effective ranking algorithm will tend to rank such pages higher than those that do not receive clicks. In this dissertation I am interested in the potential of log-based evaluation, and thus a relatively naive click model is sufficient for that purpose. More complex models of interaction will likely generate an improved test collection.

Sessions File Format

This dissertation aims to build a pipeline for automatic generation of test

collections which can be used across any digital library which has transactional logs available. The choice was made to output the session data into a standardised format as shown in listing 3 below. This format provides all the information required to change extraction methodologies easily. The file is a JSON formatted file, which is a common data-based exchange file format and is not application specific. This facilitates interoperability between the analysis and generation stages of the pipeline, meaning that the languages of implementation could be easily changed. This is highly important in the area of digital humanities as the architecture and technology choices made across the domain are both wide and varied.

```
{
    clientip: [
        {
            "clicks": {
                click_id: {
                    "artefact": url,
                    "timestamp": timestamp
                },
                ...
            },
            "queries": [ query ],
            "start_time": timestamp,
            "user": clientip
        },

        ...
    ],

    ...
}
```

Listing 3: Basic Structure of Sessions JSON File

3.6 Analysis

3.6.1 Overview

The next step in the TLA pipeline is the analysis stage. The purpose of this stage is to evaluate the performance of the transactional log generated test collection against what is expected. Each query from the set of queries will be issued against the document collection and the top k documents are returned from each ranking algorithm, where k is chosen to be 20 for this dissertation. By comparing the overall performance of the ranking algorithm to each other, this dissertation aims to ascertain if their trend follow what is given in literature.

Outlined in section 3.6.3 below are the ranking algorithms which have been chosen. These well known ranking algorithms include both state of the art and classical algorithms. The purpose of these ranking algorithms is for comparison, to determine that the test collection performs as expected on each of these algorithms. A number of performance metrics are used to determine the effectiveness of each algorithm on the test collection, and are described in Chapter 5.

How to search across documents?
Now that we have relevancy judgments and queries, the document collection upon which the test collection is generated must be indexed. By indexing the documents, they can be searched across using various ranking algorithms. I decided that a suitable application which could provide this functionality is Apache Lucene. The following section discusses Lucene and why I chose it.

3.6.2 Lucene Application Design

Lucene provides an interface for both indexing and searching across documents. The design decision was made to interact with Lucene directly, instead of using a wrapper application such as Apache Solr. Apache Solr abstracts away the details of Lucene over a HTTP API interface. Although this was appealing, it did not

allow for manipulation of the similarity ranking algorithms. With Lucene I was able to search across the indexed document collection using different ranking algorithms, as outlined in section 3.6.3. This decision led to a lower-level of interaction with the Lucene Java API, however it's benefits outweighed the costs as outlined above.

Outlined in figure 3.6.1 is the process by which Lucene indexes and searches across documents. A query is provided to the Lucene searching mechanism, which then uses the built index of documents to compute relevancy scores. This relates back to the foundation of Information Retrieval, where a document is given a relevancy score based on some criteria.

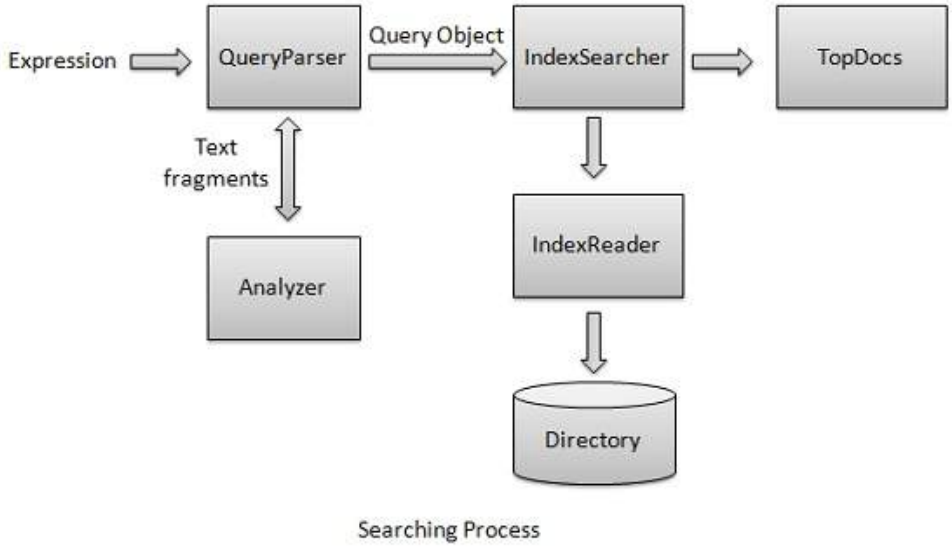

Figure 3.6.1: Lucene Search Process

`https://www.tutorialspoint.com/lucene/images/searching_process.jpg`

3.6.3 Ranking Algorithms

Outlined in this section are the three ranking algorithms selected for evaluation of the test collection. These algorithms are available for usage in the Lucene environment and are implemented slightly differently from the theoretical

formulae outlined below, however the differences are minimal and do not detract from the general purpose of the algorithms.

These algorithms were chosen as a means to evaluate the effectiveness of the test collection against what is expected. A baseline ranking algorithm, namely the Vector Space Model will be benchmarked against other ranking algorithms such as BM25, which is a state-of-the-art ranking algorithm.

Vector Space Model

$$tf - idf_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t} \tag{3.1}$$

The Vector Space Model is a classic ranking algorithm when it comes to information retrieval, it was the default in lucene search for many years and has only recently been replaced by BM25, which is discussed later.

How does it work?
The Vector Space Model is based on Term Frequency - Inverse Document Frequency (TF-IDF). TF-IDF weights values based on the product of the occurrence of words in text and the inverse of the occurrence of these words across documents.

Term frequency $tf(t, d)$ is the frequency of a term in a document, which is the number of times that term t occurs in document d.

The inverse document frequency (IDF) component is a measure of whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

Combining these two components gives term frequency inverse document frequency (TF-IDF). A high value of TF-IDF is reached by a high term frequency and a low document frequency of the term in the whole collection of documents.

Jelinek-Mercer Smoothing

$$p_\lambda(w|d) = (1-\lambda)p_{ml}(w|d) + \lambda p(w|C)) \tag{3.2}$$

Jelinek-Mercer smoothing is a *language model*, which has a single parameter, λ, which affects the influence of each model. The optimal value depends on both the collection and the query. The optimal value is around 0.1 for title queries and 0.7 for long queries.

How does it work?
A language model is a probability distribution over strings $P(s)$ that attempts to reflect the frequency with which each string s occurs as a sentence in natural text. Documents are ranked by the probability that the query text could be generated by the document language model. In other words, we calculate the probability that we could pull the query words out of the bucket of words representing the document. This is a model of topical relevance, in the sense that the probability of query generation is the measure of how likely it is that a document is about the same topic as the query.

The major problem with this estimate is that if any of the query words are missing from the document, the score given by the query likelihood model for $P(w|d)$ will be zero. Smoothing is a technique for avoiding this estimation problem. Typically there does not exist large amounts of text to use for the language model probability estimates. The general approach to smoothing is to lower (or discount) the probability estimates for words that are seen in the document text, and assign the leftover probability to the estimates for the words that are not seen in the text. This dissertation uses the smoothing model 'Jelinek-Mercer' as described by Zhai, [Zhai and Lafferty, 2001].

BM25

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})} \tag{3.3}$$

where $f(q_i, D)$ is q_i's term frequency in the document D, $|D|$ is the length of the document D in words, and avgdl is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters.

How does it work?
BM25 is based on a bag-of-words approach. The score of a document D given a query Q which contains the words $q_1, \ldots, q_n$ is given above. BM25 that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document. It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters, [Robertson and Zaragoza, 2009].

3.7 Summary

This chapter discussed the main design methodologies of this dissertation, relating closely to the TLA pipeline.

- **Overview of a Test Collection** - Methodology for generating a test collection via transaction logs.

  In this section an introduction to what a test collection is comprised of and the design methodology of this dissertation in generating such a collection through the usage of transaction logs, was discussed. A discussion around the design of the development stack was given, and how this dissertation was designed to extract the required information and evaluate its performance.

- **Data Extraction & Ingestion** - Collection of transaction logs from the 1916 and 1641 collections from the Cultura Project, and ingestion of the

transaction logs into a data-store.

In this section the motivation behind the design decision to aggregate and filter the transaction logs, was discussed. The approach of using a real-world information retrieval system, namely the Cultura Project was discussed and the main properties of transaction logs. Finally, the ethical implications around security and privacy were highlighted and discussed.

- **Session Generation** - Extraction of the relevant information for building a test collection.

  This section discussed the design behind generation of sessions for creating relevance judgments and the usage of a standardised sessions file and how this format facilitates interchange of different extraction methodologies.

- **Analysis** - Running multiple ranking algorithms over the indexed collection.

  Outlined in this section were the chosen ranking algorithms and a brief outline of their behaviours. This included a discussion on the design of the Lucene-based application, and why Lucene was chosen to evaluate the generated test collection.

4

Implementation

4.1 Data Extraction & Ingestion

Chapter 3 discusses the motivation behind the choice of Elasticsearch and Logstash to extract and store the transaction log data. It's scalability and platform independence were significant factors for the large datasets used by this dissertation.

In the final version of the system the following requirements, set out during the design phase, were implemented:

1. The system should be sufficiently flexible to allow different log formats.
2. The system should account for time zone differences in transaction logs, to simplify the querying process and make the approach transferable to transaction logs in other timezones.

3. The data stored should be easily queryable and require minimal support on a client.

The first requirement is the extraction of data from raw transaction log files and a requirement for the system to be flexible enough to work on different log formats. This involves using Logstash, an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to a datastore, namely Elasticsearch in this dissertation.

Logstash uses regular expressions to parse files and extract information from them. Regular expressions allow the design of such a solution to change the way that information is extracted from a file. Shown in listing 4 is the regular expression defined for extracting standard Apache log format files. This looks for patterns which match the expression, and assigns a field name to the matching section. For example *IPORHOST:clientip*, looks for a pattern matching a standard IPv4 address, extracts it and assigns the field name 'clientip'.

```
1  COMMONAPACHELOG %{IPORHOST:clientip} %{USER:ident} %{USER:auth}
2      \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}
3      (?:HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})"
4      %{NUMBER:response} (?:%{NUMBER:bytes}|-)
5
6  COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}
```

Listing 4: Regex Pattern for Apache Logs

With Logstash's pattern recognition defined, the next step is to account for time zone differences in transaction logs. Outlined in Listing 5 is the configuration file which provides Logstash with information it needs to find the logs, the pattern to match and finally where to output the results. As can be seen in this example, a transformation is defined from lines 11-13. The time stamp given by the logs can be from any time zone, and therefore needs to be changed

to a standardised zone, allowing for querying across the dataset. This means that timestamps which occur at the same time, but in different time zones are converted to GMT, allowing any query to correctly assume these times are the same.

```
input {
  ...
}

filter {
  grok  {
      patterns_dir => ["/etc/logstash/conf.d/patterns"]
      match => { "message" => "%{COMBINEDAPACHELOG}" }
   }
  date {
    match => ["timestamp", "dd/MMM/YYYY:HH:mm:ss Z"]
    locale => "en"
    target => "@timestamp"
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "cultura"
  }
}
```

Listing 5: Configuration file - Logstash

With Logstash now setup and transforming the transaction logs into a usable format, the next step is to setup the Elasticsearch node which will store this information. A service is defined which will setup and run Elasticsearch.

With the Elasticsearch instance running and with data now available for usage,

how do we access the data? Of key importance in the design of the system was the data being easily accessible. Elasticsearch provides a HTTP based API which allows for standard HTTP requests to be issued against it. This means that the implementation of any client accessing the resources are language independent and only require being able to issue HTTP requests. Elasticsearch also has it's own query language which provides a means to declare powerful and expressive queries across an index. An example of the implemented system is shown in figure 4.1.1, with a simple query asking for all documents where the accessed resource contains the word *1641*.

```
neil@vma11:~$ curl -XGET "http://localhost:9200/cultura/_search?pretty&size=1" -
[d'                                                                            ]
{
    "query": {
        "match": {
            "request": "1641"
        }
    }
}'
  "hits" : {
    "total" : 461186,
    "max_score" : 2.0157952,
    "hits" : [ {
      "_index" : "cultura",
      "_type" : "logs",
      "_id" : "AVrw5dlqxZ3pNEIuOZGH",
      "_score" : 2.0157952,
      "_source" : {
        "message" : "37.228.213.63 - - [13/Jul/2013:19:56:49 +0000] \"GET /1641/
 HTTP/1.1\" 200 4905 \"http://cultura-project.eu/\" \"Mozilla/5.0 (Macintosh; In
tel Mac OS X 10_8_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.71
Safari/537.36\"",
        "@version" : "1",
        "@timestamp" : "2013-07-13T19:56:49.000Z",
        "path" :
        "host" :
        "clientip" :
        "ident" : "-",
        "auth" : "-",
        "timestamp" : "13/Jul/2013:19:56:49 +0000",
        "verb" : "GET"
        "request" : "/1641/",
        "httpversion" : "1.1",
        "response" : 200,
        "bytes" : 4905,
        "referrer" : "\"http://cultura-project.eu/\"",
        "agent" : "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/28.0.1500.71 Safari/537.36\""
      }
    } ]
  }
}
```

Figure 4.1.1: Elasticsearch Query

This concludes all the necessary implementation to provide the next stage of

the pipeline with the information necessary to generate sessions. The solution outlined is both flexible and scalable, key factors in the choice of Elasticsearch and Logstash over using other solutions such as SQL. SQL provides the ability to query the dataset, however it's performance degrades dramatically when issuing queries over large datasets. The system designed achieved all of the requirements set out at the design stage.

4.2 Session Generation

The next step in the designed solution was to generate sessions from the transaction logs. This requires the development of an application which will provide this functionality.

The motivation behind generating sessions is that it would provide the relevance judgments which are required to create a test collection. Therefore the application needed to be able to carry out some specific functionalities for this purpose:

1. Provide a means to query Elasticsearch and import the data as returned by it's API.

2. Generate sessions according to the methodology defined in the design process.

3. Export the data into the defined JSON format structure.

It was chosen to create this application using the *Python* programming language. Python provides a number of libraries which manipulate timestamps along with built-in JSON parsing and export functionality. Python also allows querying over the dataset using generators. Generators are a feature of python which reduces the memory usage footprint of the application by bringing data into memory as needed. Bringing such large datasets into memory is not feasible, therefore the usage of generators is important.

In order to achieve the first requirement of this application, the official python library for Elasticsearch, *elasticsearch-py*[1], was added. This library provides functionality around connecting to and querying against an Elasticsearch cluster. Once added, a connection is then made to the Elasticsearch cluster containing the transaction log data, as shown on line 3 of listing 7 below.

```
1 from elasticsearch import Elasticsearch, helpers
2
3 class ESHelper():
4     def search_es(self, index="", body={}):
5         return helpers.scan(self.es, index=index, scroll='2m',
          ↪  query=body)
```

Listing 6: ESHelper.py

```
1 from es_helper import ESHelper
2
3 es = Elasticsearch([{'host': 'localhost', 'port': 9200}])
4
5 def search_all_queries(choice=""):
6     return es_helper.search_es("cultura", {"query": { "bool": {
       ↪  "must": [
7         { "match": { "request": choice } },
8         { "match": { "request": "searchQuery" } }]}}
9     })
```

Listing 7: Extract for Querying Elasticsearch

The excerpt above requests all data containing a search query. The resulting dataset is now available, however this is not brought into memory by the

[1] https://elasticsearch-py.readthedocs.io/en/master/

application, as the data is quite sizeable. A generator is used to iterate over this, bringing the data into memory when needed.

The next step in the implementation of this experiment is to generate the sessions using my language of choice, python. During the design phase of this experiment, the definition of a session and how this experiment represents it is shown in Chapter 3, section 3.2.3. Outlined in Listing 8 below is the implementation of the designed session-based segregation of data from the transaction logs.

```
1  data = search_all_queries(choice)
2
3  for entry in data:
4      ...
5
6      if query:
7          startTimestamp = parse(timestamp)
8          endTimestamp = startTimestamp + timedelta(hours=1)
9
10         # check for other queries by the same user, within an hour
11         is_other_queries = check_other_queries(startTimestamp,
           ↪  endTimestamp, client_ip, choice)
12         ...
13
14         click_data = search_clicks(startTimestamp, endTimestamp,
           ↪  client_ip, choice)
15         new_session = start_new_session(startTimestamp, query,
           ↪  client_ip, click_data=click_data)
16
17         sessions[client_ip].append(new_session)
```

Listing 8: Implementation of a Session

Listing 9 below shows the implementation of the defined function *check_other_queries*, which provides the functionality of retrieving all clicks by a

particular user within an hour of some query. This illustrates an example of the query language as used by Elasticsearch, which provides powerful, complex filtering on the Elasticsearch cluster.

```
def check_other_queries(timestamp1, timestamp2, clientip, indx):
    return es_helper.search_es("cultura",
        {
           "query": {
                "bool" : {
                    "must": [
                        {"range" : {
                            "@timestamp" : {
                                "gte": timestamp1_str,
                                "lt": timestamp2_str,
                                "format": "dd/MMM/YYYY:HH:mm:ss"
                            }
                        }},
                        {"match": {
                            "clientip": clientip
                        }},
                        {"match_phrase": {
                            "request": "? q = artefact"
                        }},
                        {"match": {"request": indx}}
                    ],
                }
           }
        })
```

Listing 9: Elasticsearch Query for Session Generation

The final stage in the pipeline stage of generating sessions is to output the result of this process to a file, so the final analysis stage of the TLA pipeline can be achieved. Included in the python basic libraries exists the functionality to export

data structures to JSON formatted files. This process is shown in Listing 10 below. The structure of this file is described in Chapter 3, and the reason why this format was chosen.

```
1 with open('all_queries.json'.format(choice), 'w+') as f:
2     json.dump(set_of_queries, f, sort_keys=True,
3         indent=4, separators=(',',': '))
4     f.write('\n')
```

Listing 10: JSON export from Python

Overall, the application was implemented using the python programming language, as it provided all the functionality required to generate sessions using a minimal number of extra dependencies (libraries). The application achieved all of the requirements as set out at the design stage, and is easily extended to other session design models as set out for future work.

4.3 Analysis

The final step in the designed solution was to *index* the document collection, and to take each query from the transaction logs and run it against this indexed collection to deem the top 20 relevant documents as per the ranking functions described during the design phase of this experiment. This requires the development of an application which will provide this functionality.

The motivation behind this stage of the implementation is to complete the last step towards creation of the test collection itself, provide all the relevant data which is needed to evaluate a test collection, and to reason about the performance of the generated test collection against what is expected. The comparison of the ranking functions and the relevance judgments as generated from the previous section is detailed in the following chapter. This application therefore needed to be able to carry out some specific functionalities for this purpose:

1. Index the document collection, so it can be searched over.

2. Parse the sessions file and extract information from the sessions as per the methodology described during the design phase.

3. The automatically generated relevance judgments and the ranked documents according to each of the ranking functions for each query are formatted into the TREC format, which will provide automatic generation of *performance metrics*. Performance metrics provide an evaluation using various techniques, as described in Chapter 6, of how well a set of relevance judgments perform in relation to a ranked set of judgments generated by a ranking function. As discussed previously, the metrics for the context of this dissertations are used to assess the rank between different IR models against those reported in literature.

The programming language of choice was *Java* for this stage of the implementation. Lucene [2] is a java-based application and provides its libraries in Java. These libraries provide the low-level interaction with the lucene API which is required by this experiment, so the ranking functions can be changed. Listing 11 below shows the functionality of the application which changes the ranking function to one of those as described during the design phase in Chapter 3. *Note:* the ClassicSimilarity class which is shown on line 11 refers to the Vector Space Model.

[2] `https://lucene.apache.org/`

```
switch(query_type) {
    case BM25:
        config.setSimilarity(new BM25Similarity());
        break;
    case SMOOTH:
        config.setSimilarity(
            new LMJelinekMercerSimilarity(SMOOTH_FACTOR)
        );
        break;
    case CLASSIC:
        config.setSimilarity(new ClassicSimilarity());
        break;
    default:
        break;
}
```

Listing 11: Lucene Similarity Configuration

The second requirement for this stage of the implementation is parsing of the sessions file which was generated, and to create the Union Unique set of relevance judgments as described in the design phase.

Indexing is a process whereby a document is represented in a way which allows it be searched across. The process by which this happens is not the focus of this experiment, merely a way to search over the documents. Once the document collection has been indexed, each query must be issued against it using each ranking function. The process of searching is described in Chapter 3, and is implemented in this application, as shown in Listing 12. Each query from the set of queries is parsed using a *Standard Analyzer*, which provides grammar based tokenisation and works well for most languages. The query is also *escaped*, whereby it reduces ambiguity in quotes (and other characters) used in that string. An example of an analysed piece of text is given as follows:

"The 2 QUICK Brown-Foxes jumped over the lazy dog's bone."

which creates a number of terms:

[the, 2, quick, brown, foxes, jumped, over, the, lazy, dog's, bone]

```
1  Directory directory = FSDirectory.open(Paths.get(INDEX_DIRECTORY));
2  DirectoryReader ireader = DirectoryReader.open(directory);
3  IndexSearcher isearcher = new IndexSearcher(ireader);
4  Analyzer analyzer = new StandardAnalyzer();
5
6  QueryParser queryParser = new QueryParser(FIELD_CONTENTS, analyzer);
7  Query query;
8  try {
9      query = queryParser.parse(QueryParser.escape(searchString));
10 }
11 catch (ParseException p) {
12     // nothing we can do if we cant even parse the query
13     return;
14 }
```

Listing 12: Excerpt of Query Parsing in Lucene

```
Indexed 8129 documents.
Trumray, [830284r194,]
Tooles Cuntry , [811135r100,]
Fermanagh, [813204r142,]
Carnyes land, [826140r155,824013r019,826143r160,]
Castle of Kilmore, [827248r220,]
River of Eny, [817213r169,]
betweene Barnerea & Bally Lunila , [824043r047,]
gerald relie, [809060r028,]
bandon, [827145r155,]
```

Figure 4.3.1: Console Output of Relevance Judgments

The final stage of analysis is to export the relevance judgments from the transaction logs and those from the ranking functions, into a the format required

by TREC's automatic performance evaluation software (trec_eval) [3]. This is the standard tool used by the TREC community for evaluating ad-hoc retrieval experiments, given the results file and a standard set of judged result from the experiment carried out in this dissertation. In this experiment, the results file is the output from each ranking function, and the judged result set is the binary relevance judgments as generated by the transaction logs through the automatic process described in this dissertation. This evaluation software requires the judgments to be formatted as shown below.

TOPIC ITERATION DOCUMENT # RELEVANCY

- *Topic* is the topic number, or query number, which identifies the query.
- *Iteration* is the feedback iteration (almost always zero and not used).
- *Document #* is the identifier of a document.
- *Relevancy* is a binary code of 0 for not relevant and 1 for relevant.

Figure 4.3.2 below shows the resulting file which is generated by this application for the Cultura Project 1641 collection.

```
0 0 831273r203 1
1 0 835213r247 1
2 0 815243r317 1
2 0 836002r003 1
2 0 812319r260 1
2 0 812050r064 1
2 0 813380v312 1
2 0 833020r016 1
2 0 809266r154a 1
2 0 809301r188 1
2 0 836177r083 1
2 0 816219r136 1
```

Figure 4.3.2: TREC formatted output - Binary Relevances

[3] http://trec.nist.gov/trec_eval/

The pipeline for automatic generation of a test collection as described previously, uses TREC software for generating the comparison of performance. Various performance metrics are used to determine how well the relevance judgments compare to the ranking functions' results.

4.4 Summary

Throughout this chapter the implementation of the components which were involved in the TLA pipeline for this experiment, were discussed. This included the development of a number of applications which encompassed the collection, preparation and analysis of the resulting test collection. Each of the developed applications were structured according to the requirements as set out in Chapter 3. The following is a brief summary of the implementations described within this chapter:

1. **Data Extraction & Ingestion**
   The ingestion of transaction logs into the Elasticsearch datastore was first to be implemented. This was developed, using *Elasticsearch* and *Logstash*. The purpose of which was to collect and split the content of the logs into queryable entities. Privacy and security were two concerns which were addressed in the design of the system, along with the scalability of the proposed infrastructure.

2. **Session Generation**
   This application was developed using *Python*, with the purposes of generating sessions from the transaction logs stored in Elasticsearch. The key aspect of this stage was to provide the analysis stage of the TLA pipeline with the required information needed to model the relevancy judgments and queries to evaluate the proposed methodology of generating a test collection from transaction logs.

3. **Analysis**
   The final application developed was the Lucene, *Java* based

implementation which would provide the functionality of indexing the document collection and formatting both the generated relevancy judgments from the sessions and the documents which each ranking function deemed relevant to each query. This was implemented in Lucene to interface with the Java API, which allowed the ranking function to be changed.

"Do something. Get results. Decide if it's worth repeating. If the results don't scream: 'Do this again!' Try something new."

- Tony Robbins

5 Evaluation

The following sections are an evaluation of the automatically generated test collection from transaction logs. The purpose of the evaluation is to compare the overall performance of the ranking algorithms to each other, with the aim to ascertain if their trend follows what is given in literature. The following sections outline the methodology / procedure for evaluating the generated test collection, and a discussion of the results of the experiment.

5.1 Method

Jelinek-Mercer Smoothing > BM25 > Vector-Space Model

Figure 5.1.1: Ordered Ranking of the Ranking Functions

The method for evaluating the generated test collection using the TLA pipeline as discussed in the previous chapters, was to compare the performance of the ranking functions against each other to determine if they conform to my expectations, as above in figure 5.1.1. This shows the expected performance ranking of the chosen ranking functions as described in Chapter 3. This requires the following steps:

1. Take each query from the generated set of queries, and search across the document collection, using each of the ranking functions chosen. This will return the top 20 documents as deemed relevant by the ranking function for that query.

2. For each query, compare (step 3) the automatically generated binary relevance judgments to the output of these ranking functions from step 1 above.

3. A number of performance metrics are selected and used to compare the results. Section 5.2 below outlines the metrics chosen and their usage in evaluating relevancy judgments.

5.2 Procedure

The procedure for evaluating the performance of a set of relevancy judgments, involves using appropriate performance metrics to measure their effectiveness.

This dissertation uses common, popular amongst the TREC community, performance metrics. These are outlined as follows:

1. **Mean Average Precision**

$$MAP\,(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{M_j}\sum_{k=1}^{M_j} Precision(R_{jk}) \tag{5.1}$$

Where the set of relevant documents for an information need $q_j \in Q$ is $\{d_1, \ldots d_{m_j}\}$, M_j is the mean for query j, $|Q|$ is the total number of queries

and R_{jk} is the set of ranked retrieval results from the top result until you get to document d_k.

2. **Mean Reciprocal Rank**

$$MRR\,(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{rank_i} \tag{5.2}$$

Where rank(j) is the position of the relevant result in the j'th query and |Q| is the total number of queries.

3. **Normalised Discounted Cumulative Gain**

$$NDCG\,(Q,k) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}Z_{kj}\sum_{m=1}^{k}\frac{2^{R(j,m)}-1}{\log_2(1+m)'} \tag{5.3}$$

Where Z_{kj} is a normalisation factor calculated to make it so that a perfect ranking's NDCG at k for query j is 1, and $R(j,m)$ is the relevant document at (m,j). For queries for which $k' < k$ documents are retrieved, the last summation is done up to k'.

4. **Precision @ 10**

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|} \tag{5.4}$$

Precision is the fraction of the documents retrieved that are relevant to the user's information need. Precision @ k defines a cut-off whereby the top k results, where k is 10 in this case, are used instead of all results.

5. **Success @ 10**
Success @ 10 is a simple metric, defined as the proportion of queries where the correct answer appears in the top k, where k is 10 in this case, [Craswell and Hawking, 2005].

	Success @ 10	**Precision @ 10**	**MRR**	**MAP**	**NDCG**
BM25	0.7183	0.1018	0.5512	0.4384	0.5063
Smoothing (0.9)	0.7218	0.1014	0.5645	0.445	0.5137
Vector Space Model (baseline)	0.7465	0.1056	0.5809	0.459	0.5285
Smoothing (0.5)	0.7394	0.1042	0.5867	0.4586	0.5284
Smoothing (0.1)	**0.7535**	**0.107**	**0.5914**	**0.4626**	**0.5323**

Table 5.3.1: 1641 Collection Results

	Success @ 10	**Precision @ 10**	**MRR**	**MAP**	**NDCG**
Smoothing (0.9)	0.2948	0.0675	0.1505	0.0477	0.0995
Vector Space Model (baseline)	0.2976	0.0686	0.1508	0.0483	0.1
Smoothing (0.5)	0.2955	0.068	0.1516	0.0485	0.1003
Smoothing (0.1)	0.2989	0.0687	0.1516	0.0486	0.1006
BM25	**0.2991**	**0.0696**	**0.1528**	**0.0486**	**0.1007**

Table 5.3.2: 1916 Collection Results

5.3 Results & Observations

Tables 5.3.1 and 5.3.2 report the results of the evaluation conducted with trec-eval on both the 1916 and 1641 collections.

1641 Collection

The 1641 collection did not perform as expected, with BM25 performing worst, followed by the vector space model and a low valued smoothing performing best, across the metrics.

Looking at the MRR metric, it is noted that the differences between BM25 to Smoothing (0.1) are modest, but these differences are *significant* (one-tailed) using the Paired Samples T-test. This is true for BM25 against both the Vector Space Model and Smoothing (0.1) on a 99% confidence level for the MRR score. This signifies that there is an underlying reason why the algorithms performed different on the collection, and that the results are not noise or randomly

achieved.

The content of the 1641 collection is transcribed from witness statements, and this translation could be a contributing factor for the performance of the ranking functions on this collection. Transcribing documents is a process whereby a human / machine will attempt to read the text of a hand written document and digitise it. It has been the subject of much research, [Siegler and Withrock, 1999], with the main issue of transcribed documents being that there exists a rate of error on the spelling of words from text to machine. This error can in fact result in the words being searched for by a user to be incorrectly labelled as (non) existent for a document. In the case of 1641, the spelling errors are not just transcription errors, but also artefacts of the form of English used at the time of writing. Some of these errors are intentional, and so this poses a problem for relevancy using ranking functions. The issued queries by users will be in modern English and may receive no resulting documents due to this language difference. Although there are techniques to reduce this, such as stemming and lemmatization [Manning et al., 2008], the problem is quite severe in the 1641 collection.

The results achieved for the 1641 collection are similar to those by Kamps [Arampatzis et al., 2007]. The values are slightly lower than those of Kamps for the Union Set, however the ranking in this experiment was not as expected.

1916 Collection

The 1916 collection performed as expected with the vector space model performing worse than BM25, and smoothing performing slightly better than the Vector Space Model, however Smoothing did not surpass BM25. As can be seen from table 5.3.2 above, the differences were very small between the values for each metric. Across all metrics BM25 performed best.

Looking at the MRR metric, it is noted that the differences between Smoothing (0.9) to BM25 are very small, and that these differences are not significant. This signifies that the resulting rankings were relatively random, and that there does not appear to be an underlying difference in their performance. This suggests that the 1916 collection is not an ideal candidate for the approach

taken in this experiment.

The 1916 collection contains a large number of queries for a small amount of documents. This could be a factor in the resulting low precision values as shown in the table above. It has been observed that precision increases with collection size. One explanation could be that the redundancy of information increases, making it easier to find multiple documents conveying the same information, [Koolen and Kamps, 2010].

5.4 Summary

In summary, the results for 1916 are mostly in line with expectations set beforehand, namely that Vector Space Model would be the worst-performing system, with both Smoothing and BM25 improving over the Vector Space Model, however there does not exist a significant difference between the ranking function, showing that the users expectation is different to the reality of the collections content.

The 1641 collection results were not as expected, with BM25 performing the worst, and Smoothing performing the best. The differences are modest but significant, which is very interesting. This could potentially be due to the nature of the collection contents, the document length of the 1641 collection is quite large and this generally leads to poorer performance of BM25, [Lv and Zhai, 2011].

6

Conclusion

The goal of this dissertation is to determine whether user clicks can be interpreted as absolute judgments of relevance, using transaction logs to extract these judgments, for the domain of Digital Humanities. By extension, whether or not these clicks can be used to generate an evaluation dataset. The proposed experiment involved the implementation of a transaction log analysis approach for extracting relevance judgments from transaction logs. The purpose and goal of this dissertation was set out in Chapter 1, and the conclusions on completion are as follows:

1. **Collection**
   In order to evaluate the feasibility of the generated test collection, it was necessary to acquire a dataset of real-world transaction logs from an active digital collection. As discussed in section 6.1 below, this led to the use of transaction logs from the Cultura Project.

These transaction logs needed to be made useful, by filtering each line and making it available for analysis through a datastore. For this purpose, an *Elasticsearch* and *Logstash* architecture was developed to filter and store this data. The designed architecture performed well in practice, catering for the large quantity of transaction logs. It also fulfilled the security expectations of an application storing user interactions. Great care was given to the extensibility of the architecture, which could cater for different log formats and timezone transference. A flaw in this design however is the setup costs associated with it. Although the architecture is built on open-source software and is operating system independent, it requires knowledge to setup and maintain. Another potential criticism is that Elasticsearch by default provides no security. A large number of Elasticsearch clusters are setup on the public facing web, with no security at all which can lead to vulnerabilities, [Connoly, 2017]. This is an inherent flaw in the system itself, but one which is quite important to consider whilst transferring the approach to other collections.

2. **Preparation**

   The most important aspect of a test collection is the relevance judgment set. In order to generate relevance judgments (from transaction logs) for a query, an approach of separating each user interaction with the system into sessions was taken. Each session represents a users information need and the documents they clicked which appear to satisfy that need. The designed representation of a session was quite simple, and could be a factor in the lack of statistical significance between the ranking functions for the 1916 collection.

   The main critique around the generation of relevance judgments is that the usage of session boundaries may not lead to a good test collection. Future work should extend the session design beyond using timeout boundaries. This may lead to a better performing test collection, as the resulting judgments of relevance may be more appropriate than using a simple one

hour boundary.

3. **Analysis**
   To ascertain the viability of the approach of using transaction logs to generate a test collection, the performance of a number of ranking functions is evaluated, with respect to one another. This evaluation is not an evaluation of the ranking functions themselves, rather a comparison of their performance on the test collection.

   The evaluation was performed on multiple collections, to ascertain the transferability to some extent of the approach. The performance metrics as described in Chapter 5, were used to compare the two collections. A round of statistical significance testing was carried out which shows whether the difference in performance of each ranking function is significant. The results of the 1641 collection are statistically significant, however the ranking functions did not perform in the order as expected. This could be due to the nature of the 1641 collection itself, and is the focus for future work. This discrepancy in the expected results is very interesting and illustrates the challenges of building such a dedicated collection which performs well on various collections. The 1916 collection performed as expected, however the results were not statistically significant. As mentioned in Chapter 5, this could be due to the nature of the collection, being such a small amount of documents for a large number of queries. Future work aims to use techniques such as random sampling as outlined by Zhang, [Zhang and Kamps, 2010].

6.1 Challenges of Digital Collections

This dissertation set out to investigate the suitability of binary relevance's deemed from transaction logs using the Trinity College Dublin Digital Collection as a use case. This digital collection provides historical documents including manuscripts, early printed books, and a number of general collections.

This digital collection however proved impossible to apply the techniques outlined in this dissertation for generating a test collection. The transaction logs and user interface naming conventions proved inconsistent, with document identification unreliable via transaction logs. This illustrates the difficulties involved in the area of information retrieval within digital humanities, showing that archives and collections vary greatly depending on the implementation and infrastructure under-pining their usage.

6.2 Final Thoughts

Overall, this dissertation has looked at three sets of transaction logs, and found the results to be unsatisfactory in each case. For the initial Trinity College Dublin Digital Collection, the log files were too noisy and unsuitable, the 1641 depositions did not perform not as expected and finally the 1916 collection resulted in a statistically insignificant ranking.

The techniques used for generating a test collection from transaction logs have been described and evaluated throughout the experiment, alongside a set of alternative approaches used in state-of-the-art. Further work and areas of improvement have been discussed to improve the effectiveness of the automatically generated test collection.

The results of this experiment should not be interpreted as a claim that the general approach of using TLA is not viable, rather that the collections used by this experiment proved unsuitable for the approach.

Bibliography

[Arampatzis et al., 2007] Arampatzis, A., Kamps, J., and Nussbaum, M. K. N. (2007). Deriving a domain specific test collection from a query log.

[Belkin et al., 1995] Belkin, N. J., Cool, C., Stein, A., and Thiel, U. (1995). Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *EXPERT SYSTEMS WITH APPLICATIONS*, 9:379–395.

[Bennett et al., 2007] Bennett, G., Scholer, F., and Uitdenbogerd, A. (2007). A comparative study of probabilistic and language models for information retrieval. In *Proceedings of the Nineteenth Conference on Australasian Database - Volume 75*, ADC '08, pages 65–74, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Borgman, 1996] Borgman, C. L. (1996). Why are online catalogs still hard to use? *Journal of the American Society for Information Science*, 47(7):493–503.

[Connoly, 2017] Connoly, D. (2017). Elasticsearch users increasingly targets of ransomware, available at `http://www.itworldcanada.com/article/elasticsearch-users-increasingly-targets-of-ransomware/389886`. [Online; posted 16-January-2017].

[Craswell and Hawking, 2005] Craswell, N. and Hawking, D. (2005). Overview of the TREC-2004 Web Track, available at `http://trec.nist.gov/pubs/trec12/papers/WEB.OVERVIEW.pdf`.

[Jansen, 2006] Jansen, B. (2006). Search log analysis: What it is, what's been done, how to do it. *Library and Information Science Research*, 28(3):407–432.

[Jones and Klinkner, 2008] Jones, R. and Klinkner, K. L. (2008). Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 699–708, New York, NY, USA. ACM.

[Koolen and Kamps, 2010] Koolen, M. and Kamps, J. (2010). The impact of collection size on relevance and diversity. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 727–728, New York, NY, USA. ACM.

[Kurth, 1993] Kurth, M. (1993). The limits and limitations of transaction log analysis. *Library Hi Tech*, 11(2):98–104.

[Lv and Zhai, 2011] Lv, Y. and Zhai, C. (2011). When documents are very long, bm25 fails! In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 1103–1104, New York, NY, USA. ACM.

[Lynch, 2002] Lynch, C. A. (2002). Digital collections, digital libraries & the digitization of cultural heritage information. *Microform & imaging review*, 31(4):131–145.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[Ritchie et al., 2006] Ritchie, A., Teufel, S., and Robertson, S. (2006). Creating a test collection for citation-based ir experiments. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 391–398, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Robertson and Zaragoza, 2009] Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

[Saracevic, 1976] Saracevic, T. (1976). Relevance: A review of the literature and a framework for thinking on the notion in information science. In *Eds.), Advances in Librarianship 6*, pages 79–138. Academic Press.

[Scholer et al., 2016] Scholer, F., Kelly, D., and Carterette, B. (2016). Information retrieval evaluation using test collections. *Information Retrieval Journal*, 19(3):225–229.

[Siegler and Withrock, 1999] Siegler, M. and Withrock, M. (1999). Improving the suitability of imperfect transcriptions for information retrieval from spoken documents. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 1, pages 505–508. IEEE.

[Susan Dumais, 2003] Susan Dumais, Thorsten Joachims, K. B. A. W. (2003). Sigir 2003 workshop report: Implicit measures of user interests and preferences.

[Voorhees and Harman, 2005] Voorhees, E. M. and Harman, D. K. (2005). *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press.

[Zhai and Lafferty, 2001] Zhai, C. and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 334–342, New York, NY, USA. ACM.

[Zhang and Kamps, 2010] Zhang, J. and Kamps, J. (2010). *A Search Log-Based Approach to Evaluation*, pages 248–260. Springer Berlin Heidelberg, Berlin, Heidelberg.