Supporting Reflection in Exploratory Search through Coordinated Visualisation

Submitted to the

University of Dublin, Trinity College

for the degree of

M.A.I. in Computer Engineering

Charlotte Cuddy

Supervisor: Prof. Owen Conlan

Submitted to the University of Dublin, Trinity College, May, 2015.

Declaration

I, Charlotte Cuddy, 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.

Signature Date

Summary

The dramatic surge in information available digitally has resulted in a move to performing research using digital collections. The volume of documentation available digitally enables thorough investigation into a subject of interest; however, exploration and ideation using large digital collections can be difficult without the support of technology to help users to extract the appropriate subset of documents that meet their needs. With exploratory search, the information needs of the user are continuously evolving as the user gains more domain knowledge and changes the focus of their query. Research has shown that this type of search requires collaboration between the user and the system, so that the technology may provide the required support, informing the user of connections of interest, while the user retains control through adjusting the system's perception of their interests to better reflect their actual interests.

Reflection is a key component in this collaboration. In order to make beneficial adjustments to the system's perception of their interests, the user must first understand their own information needs. Through self-reflection, the user can gain an understanding of the exploration path they have followed from which they may generate hypotheses or identify goals and lay out activities to perform in order to ratify their hypotheses or reach their goals. Reflection may also lead to serendipitous discoveries which may contribute to hypothesis generation, or change the direction of the exploration. Due to the large volume of textual information consumed during the exploration it is difficult to reflect on the path taken textually. Tools are required which render meaningful representations, illustrating to users how they have interacted with documents and concepts.

In this project, a framework to generate scrutable and controllable user models to support reflection, using coordinated visualisations, is developed. Coordinated visualisations facilitate the exploration of large complex datasets. Through a visual interactive interface linking different visualisations of the user model, the user can explore the behaviours of various subsets of their interests. This enables them to gain a better understanding of their exploration path, supporting self-reflection, hypothesis generation and serendipitous discoveries through scrutability.

This framework is predicated on the semantic analysis of a collection of documents and the log analysis of user interactions with resources during their exploration; data from the CULTURA 1641 collection was procured for the development of the framework.

The framework was implemented using a client-stateless server architecture. The Dimensional Charting JavaScript library, which uses the Crossfilters and the Data-Driven Documents JavaScript libraries, was used to develop the coordinated visualisations. The data used to generate these visualisations was held in MySQL databases and an XML document and was accessed using jQuery and AJAX requests through the Slim Application, a REST API.

The framework developed incorporates features recommended to assist in the exploration and analysis of large data sets. Through the successful performance of complex analysis, an evaluation of the framework demonstrated its user-friendliness and technological capability, as well as the scrutability of the user model through the use of the system. The evaluation also validated the reflective capabilities of the framework, in particular through examining the temporal evolution of the user interests. The main weakness observed was in the choice of visualisation for the user model; the obstruction of information required participants to find alternative methods of extracting the required information.

It was concluded that the goals set out were achieved, that the coordinated visualisation user models were scrutable and controllable, and that through the scrutable and controllable nature of the user models reflection was supported; however it was also demonstrated that the use of an alternative representation for the user model would enhance the benefits derived by engaging with the framework.

Abstract

Exploration and ideation using large digital collections can be difficult without the support of technology to help the user to extract the appropriate subset of documents that meet their evolving needs. Research has shown that this type of search requires collaboration between the user and the system, so that the technology may provide the required support for that user, informing them of connections of interest, while the user retains control through adjusting the system's perception of their interests to better reflect their actual interests. In order to make beneficial adjustments to the system's perception of their interests, the user must first understand their own information needs. Reflection is key in the understanding of information needs; however, due to the large volume of textual information consumed during the exploration, it is difficult to reflect on the path taken textually. Tools are required which render meaningful representations illustrating to the user how they have interacted with documents and concepts. This dissertation proposes a framework that generates scrutable and controllable user models to support reflection using coordinated visualisations. It examines the key features of scrutable user models and of visualisations which assist in the exploration and analysis of large data sets. The design and implementation of the framework are outlined. An evaluation of the developed framework and the observations made through the evaluation are described. This dissertation concludes that the coordinated visualisation representation of the user models generated are scrutable and controllable, and that the scrutable and controllable nature of the user models supports reflection. It observes, however, that the use of a different visualisation to illustrate the user model would enhance the benefits derived by engaging with the framework.

Acknowledgements

I would like to thank my supervisor Owen Conlan for his continued help, guidance and expertise without which this work would not have been possible. Furthermore, his seemingly unwavering positivity and confidence in my capabilities were a constant motivational source throughout the year.

I would like to thank my family, friends and classmates for their continued support and encouragement, for keeping me grounded when I needed it most and for providing distractions to keep my sanity intact.

I would also like to thank everyone who took time out of their busy schedules to participate in the evaluation of the project and for the valuable feedback they provided therein.

Finally, I must thank my wonderful proof readers, Claire Liuzzi, Sandra Wright, Marion Cuddy and Michael Cuddy, for taking the time to read through this dissertation with such attention to detail, but also for the unhesitating manner in which they accepted to help.

Thank you all!

Table of Contents

Table of Figures

1 Introduction

1.1 Motivation

The evolution of the World Wide Web together with emerging technologies has enabled access to expansive collections of information. The internet has become the go-to place when information is required. As information became increasingly available online, the internet was utilised to extract specific answers to specific queries. Tools have successfully been developed to remove ambiguity in queries and hone in on the answer required. As we move towards a world of open data and digitalised collections, information requirements are changing [Marchionini, 06].

Accessibility to such a wealth of information facilitates thorough exploration of particular topics at users' fingertips and hence a move towards doing research online has been observed; however performing complex information seeking tasks online introduces challenges around the relevance of recommendations made by the system [Singer et all, 11]. Tools that combine users' implicit and explicit actions to focus in on the piece of information required become inhibitive in exploratory search. In query search, users expect the search engine to discard any information that does not match their query exactly. In exploratory search, users may not know what information they require to begin with; they merely want to explore different avenues around a particular topic. Supporting tools are expected to cast a wider net when making recommendations so as to suggest related areas that may be worth exploring [Marchionini, 06].

An added complication in search is the user's expertise in a particular domain. In query search, the computer can infer some information about the user's knowledge based on their actions. Query searches are generally performed over short periods of time so that the user's knowledge of the domain and the focus of the query remain constant throughout. During exploratory search, the information needs of the user are continuously evolving as the user gains more domain knowledge. Exploratory searches can take place over long periods of time, during which users begin to require more specialised recommendations and change the focus of their query [Athukorala et al, 14].

Collaboration is therefore required between the user and the system so that the technology may provide the required support for that user. Tools should be available to present the system’s perception of the user’s interests to the user. In turn the user should have the ability to retain control through adjusting the system’s perception of their interests as required. Through this collaboration, the relevance of the recommendations will be improved, and the user will gain understanding into why certain recommendations are being made [Koenemann and Belkin, 96].

Reflection is a key component in this collaboration. In order to make beneficial adjustments to the system’s perceptions of their interest, the user must understand their own information needs. Users may haphazardly explore a topic without paying attention to the shift in the focus of their exploration. Through self-reflection, users gain an understanding of the exploration path they have followed and how their focus has changed. Having gained an understanding of their information needs, they may generate hypotheses or identify goals and lay out activities to perform in order to ratify their hypothesis or reach their goals. Reflection may also lead to serendipitous discoveries which may contribute to hypothesis generation, or change the direction of the exploration [Ferrari, 01].

Due to the large volume of textual information consumed during the exploration it is difficult to reflect on the path taken textually. Tools are required which render meaningful representations illustrating to users how they have interacted with documents and concepts [Gauch et al., 07].

Digital content may be enriched with metadata through semantic uplift processes such as entity and relationship extraction. This metadata describing the documents is used to illustrate relationships between entities and is used in search systems and in recommender systems to identify documents containing sets of metadata. Recommender systems use metadata contained in documents interacted with by a user to build a model of the user's interests. At each user interaction, the metadata is added to the model, weighted based on the user's action. For example, metadata contained in a selection of text annotated by a user will be weighted more heavily than metadata in a document visited by a user. Visualisations can be created using the aggregation of the weighted metadata to illustrate to users what the system's perception of their interests is; however a visualisation of a user's interests at a particular point in time provides no information about why concepts are perceived as important or why some concepts are considered more important than others, nor does it support reflection of the exploration path to date. To fully understand the system's perception of their interests and how their interests have deviated over time, users must reflect over how the model has evolved throughout the exploration [Gauch et al., 07].

Despite reducing the full textual content to metadata, the aggregation of temporal changes in data can lead to large complex sets of data. For this data to be scrutable by end users and support reflection, tools are required that can deal with the analysis of such data.

Coordinated visualisation systems are a powerful tool for complex data analysis. In coordinated visualisation systems, the data is portrayed using an assortment of data representations based on different variables from the data set or variables derived from the dataset. Through linking different visualisations of a data set, users can interact with the visualisations so as to only view a subset of the total data set. Any change in the subset of data is propagated through all of the visualisations [Shneiderman, 96].

For example, the Nasdaq 100 Index over two decades could be modelled in a number of ways, aggregating the data by year, by days where a gain or a loss was observed or by yearly quarter to name but a few. By using a coordinated visualisation, a potential investor wishing to determine the best time of year to invest could interact with the visualisations to explore whether one of the yearly quarters consistently outperformed the others. Selecting the first quarter on the visualisation representing the data by quarter, the full set of data would be reduced to only consider data from the first quarter of each year under consideration and all visualisations in the coordinated visualisation would be updated.

Modelling complex data from different perspectives with the facility to interact with the visualisations allows users to explore the behaviours of various subsets of that data, enabling them to gain a better understanding of the data in order to draw conclusions about that data, illustrate relationships and identify unforeseen connections [Shneiderman, 96].

In order to develop a tool to facilitate the effective collaboration between a user and technology that supports exploratory search, the complex user model data held by the system must be transformed into visualisations from which the user can understand

- Why particular interests are present in their user model
- Why interests are perceived as having a particular importance level
- How their interests overall have changed over time
- How the importance of a particular interest has changed over time

This tool should be simple to use and should require minimal effort from the user to scrutinise the data. Through a visual interactive interface linking different visualisations of the user model, users could explore the behaviours of various subsets of their interests. This would enable them to gain a better understanding of their exploration path, supporting self-reflection, hypothesis generation and serendipitous discoveries.

1.2 Research Question

This project proposes to **develop a framework to generate scrutable and controllable user models to support reflection through coordinated visualisation.** It outlines the design and implementation of a tool with which the user can scrutinise the system's perception of their interests through a visual representation of their current user model and of previous user models from any stage of their exploration, where changes between consecutive user models are highlighted to yield further insight.

Scrutable user models are models that can be examined or inspected closely and thoroughly. They present what information is held by the system about a user. The information must be portrayed in such a way that the user can understand the information and can check the correctness of that information. The technology will take an objective approach to modelling the user's interests, processing the information collected about the user clinically. The user's actual beliefs may differ due to the subjectivity of human perception. Scrutability is required to enable the identification of any discrepancies between the user's and the system's interpretations of the user's interests. This is achieved not only by presenting the information to the user, but also by supporting the self-awareness of users. Scrutable user models will prompt users to reflect on their interest so as to clarify any ambiguity in what their interests truly are.

For the purpose of this project **controllable user models** are user models that allow user to control what information is presented in the model. To fully understand why the system perceives the user's interests to be as illustrated in the current user model, users must be able to examine previous user models. Controllable user models allow users to restrict the information presented to them to only include interactions up to an earlier point in time. Users may also throttle the granularity of the information presented so that, for example, only the dominant interests are presented. This will allow users cycle through events observing how interests came to be present in the user model and how the level of importance was determined. Through providing controllable user models, the degree of scrutability of the user model will be enhanced.

Reflection in this project refers looking back on or reviewing the exploration path that led to the generation of the current user model. This will be achieved through reviewing past user models, observing changes from one event to the next so as to gain an understanding of how their interests have evolved throughout the exploration. Through reflection, the user can benefit from self-reflection, self-regulation and serendipity. Self-reflection is the process by which a user examines what they are doing. This will allow the user to formally identify what their current interests are. Self-regulation is the generation of hypotheses or goals and the formulation of an action plan to ratify the hypothesis or achieve their goals. Serendipity is the discovery of unexpected connections, some of which will be discarded and some of which the user will incorporate into their own perception of their user model.

Coordinated visualisations are an assortment of linked visualisations of a single set of data. By interacting with the individual visualisations, users can reflect over subsets of the data. Changes to individual visualisations are propagated through to the other visualisations. By examining subsets of the data in isolation, users can reduce the volume of information presented to them so as to gain a deeper understanding about the particular subset. In particular, users can examine how the subset is affected by the variation of a single parameter, which may reveal relationships or trends which were veiled when considering all parameters in the data set.

1.3 Objectives and Goals

User models derived during exploratory search provide an interpreted view of the information requirements of users, based on the content they have interacted with and the manner in which they have interacted with it. They should provide insight into the users' interests, supporting reflection over their exploration and the evolution of their information needs. Self-reflection, self-regulation and serendipity can come out of user model reflection. Lack of scrutability and control however results in limitations in the reflective capabilities of user models. Common weaknesses observed in user models with respect to reflection are:

- There is no information, or insufficient information, regarding the rationale for the presence and importance of interests in the user model [Kay, 94].
- There is no method to view previous user models.
- There is no information about how the model has changed as a result of the current event.
- There is no information about how the model has evolved over time [Athukorala et al., 14].
- There is no method to filter the visualisation.
- There is no method to zoom in on certain aspects of the visualisation.
- There are no, or limited, details available on demand [Lum, 07].

In light of this, the project set out to overcome these challenges and produce a system that permits the analysis of previous as well the current user models using a coordinated visualisation. This would allow the user to step through the evolution of their user model, analysing changes, zooming in and filtering the data, as required, to support reflection.

In order to achieve this, the following goals were set:

- Store sufficient information about the user model at each event to be able to recreate all historical user models as well as the current user model.
- Present a user interface that is easily adopted by users to support reflection.
- Render the user model for any event highlighting changes compared to the previous event and providing useful human-readable details on demand of the changes.
- Facilitate the performance of complex analysis on the user model.
- Provide tools to view useful subsets of the model to gain a deeper insight into the user model.

- Allow the user to throttle the level of information presented to focus on primary interests only.
- Allow the user to throttle the level of information presented to analyse secondary interest so as to identify changes in the focus of their exploration and possible serendipitous relationships.
- Enable users to reflect on how the model, and hence their interests, have changed over time.

For these goals to be met, the following objectives must be realised:

- Analyse and understand the log data so as to identify the structure of the logs of the different user actions, the information required to create the user model and the most general method of extracting the required information from the log data.
- Analyse and understand the semantic data for the collection of documents on which the user actions are performed so as to identify how the log data can be used to identify the related entities and the minimal entity information required to create the user model and provide users with human-readable descriptions.
- Design the coordinate visualisation interface, identifying feasible filtering tools that will be most beneficial for users to gain insight into their user model, thus supporting reflection.
- Design the shell of the system, identifying what technologies are needed and how they should communicate.
- Design the inner working of the system, what should happen during the pre-processing of the data, during the initial rendering of the visualisation and during the user's reflection of the progress of their exploration.
- Implement the technology to meet the required goals in line with the chosen designs.
- Design an appropriate task based evaluation and test that the technology works as expected and meets the required goals.

This framework will be predicated on the semantic analysis of a collection of documents and the log analysis of user interactions with resources during their exploration. The logging of user data and the semantic uplift process are outside the scope of this project. As semantic and user log data is required to build the user models, data from the 1641 CULTURA system will be use. The CULTURA project[1] will be detailed in Chapter 2.

1.4 Overview

This chapter looked at the motivation which led to the proposal of the research question. The proposed research question was stated and the terms used in the question were explained in the context of this project. Based on how the current technology is lacking when it comes to generate scrutable and controllable user models to support reflection through coordinated visualisations, goals were set with which to measure the success of the implementation of the framework. Objectives were then set with which to meet the goals. The remainder of this chapter looks at what is covered in each chapter of this report.

Chapter 2 provides some background information on what the CULTURA project strives to achieve, challenges it faces which are relevant to this project and some of the technologies and methodologies it uses in light of these challenges.

Chapter 3 describes how information retrieval challenges have evolved due to accessibility of online knowledge and the move towards undertaking research using digital collections. It looks at what motivates scrutability and the features that scrutable visualisations should possess in order to facilitate the reflection. It considers some existing user models and identifies which of these features it does and does not have.

[1] http://cultura-project.eu/

Chapter 4 looks at the various technologies used in the implementation of the system, first on the data processing side, then on the user model visualisation side. It looks at the advantages of using these technologies, and where the technologies are lacking.

Chapter 5 describes the process undertaken to design the framework. The choice of the coordinated visualisation component, methods and technologies used in the design of the framework are explained and overall design of the system is described.

Chapter 6 looks at how the design was implemented by outlining how the semantic analysis and log analysis are combined to generate the user model data, how the user data is retrieved and stored during exploration of the user model, how the user data is converted into the visualisations, how the system deals with users moving through different events, how changes in the user model between sequential events are highlighted and how the client and server communicate.

Chapter 7 looks at the evaluation undertaken to test the final system using a task based evaluation. The ethical considerations are discussed. The findings from the evaluation are outlined illustrating where the objectives were met and highlighting areas where the system would require further development if future work were undertaken on the project.

Chapter 8 concludes that the framework was developed as per the goals set out but highlights some features that would need to be implemented for the framework to be marketable.

2 Background

This framework is predicated on the semantic analysis of a collection of documents and the log analysis of user interactions with the collection of documents. The semantic uplift process and user log recording are outside the scope of this project. Semantic and user log data from the CULTURA 1641 Depositions will be use. This chapter provides some background information on what the CULTURA project strives to achieve, the challenges it faces which are relevant to this project and some of the technologies and methodologies it uses to overcome some of these challenges.

CULTURA is a research project which aims to promote engagement with digitalised cultural heritage collections by providing an environment that supports individuals with varying levels of experience as they navigate through the collections. The CULTURA system provides tools such as annotation, visualisations, narratives and personalisation. Recommendations are made based on user actions and interests.

One of the collections the CULTURA environment works with is the 1641 Depositions, a collection of witness testimonies from Protestant men and women from all classes and from all parts of Ireland detailing their experiences around the 1641 Catholic Irish rebellion. Due to the volume of the collection and the noisiness of the text – missing words, misspellings, inconsistent grammar and sentence structure – the manual exploration of the collection has proved to be a lengthy, arduous task. By digitising the collection and providing tools to support the end user, the CULTURA environment facilitates the exploration and extraction of information from this unique source of information.

Due to the noisy unstructured nature of historical collections, including the 1641 Depositions, there was a need for the development of natural language processing tools to normalise ambiguities caused by abbreviations, spelling and punctuation inconsistencies in the collection and tools for entity and relationship extraction.

REBELS is a machine learning technique which provides ranked correction candidates for the noisy text. The likelihood of the normalisation of the noisy text is determined using a language model. The ranked candidates and likelihoods are combined to determine the best normalisation [CULTURA].

CULTURA's IBM team developed the Entity Relationship Recognition module to perform entities and the relationship extraction over a collection and output an entity graph for that collection. Persons, locations, dates and events, the relationships between them and relationships between their associated attributes were extracted from the 1641 collection using this module. To improve the accuracy of the entity and relationship extraction process, CULTURA uses Commetric's PreMapper tool for manual feedback. Access is restricted to experts, who can view and edit the entity networks, creating and merging entities and removing ambiguities as required [CULTURA].

Equalia software is used for the evaluation of the system. To perform the evaluation, Equalia collects and analyses data including log data while users interact with the system. Data is collected in XML format through a REST interface [CULTURA].

The combined use of these technologies facilitate the generation of models of users' interests which are used as a basis for personalised recommendations of activities and documents. By tracking and logging user interaction with the depositions and the entities in the depositions in the environment, the system can identify persons, events and other entities and their relationships which can then be used to build their user model and identify areas of interest. The user model can be viewed by the user to facilitate scrutability and reflection. Users can interact with the user model to adjust weights assigned to their interests in the model. This ensures transparency and gives users control, thus providing a means by which the user may collaborate with the system to enhance personalisation.

The CULTURA personalisation aims to support user at varying levels of expertise, from novice researchers to experienced researchers. To facilitate this, underpinning personalisation is a four stage personalisation approach – guide, explore, reflect and suggest – illustrated in Figure 2.1.

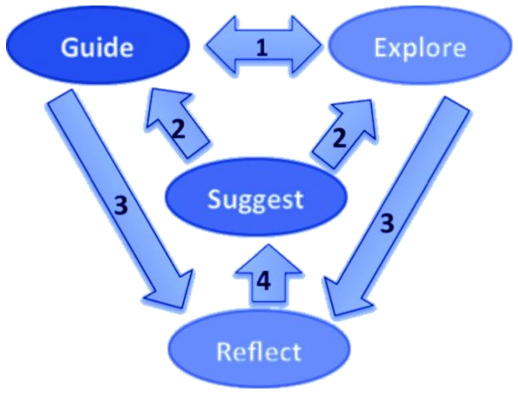


Figure 2.1: CULTURA four stage personalisation approach.

A narrative module is available for novices to guide them through a specific theme. As users gain experience and demonstrate a deeper interest in particular areas, the system provides further relevant resources. Users may also request further resources related to a particular area. Such users alternate between the guide and explore stages. The greater the experience a user gains and the deeper the understanding the user has of the collection, the more time they spend in the explore phase. Professional researchers will generally remain in the explore phase. The suggest phase works alongside the guide and explore phases. Based on the content the user is currently viewing and the user model held by the system for that user, the system makes personalised recommendations of related resources. At any stage during the guide and explore phases, the user may enter the reflect phase. In this phase, the user views their user model, reflects over the interests perceived by the system and makes appropriate changes, adjusting weight or deleting entities until the model represents their actual interests. Changes made to the user model will be incorporated by the system when making recommendations in the suggest phase [Hampson et al., 14].

This chapter looked at the technologies and methodologies used by the CULTURA project. The next chapter will look at the state of the art around frameworks used to support exploratory search which utilise user models that are scrutable and/or controllable. These frameworks will be evaluated against a set of guidelines for scrutable user models and for visualisations.

3 State of the Art

This chapter examines the challenges faced in the development of tools which support users during exploratory, and how the visualisation of user models has contributed to overcoming these challenges. Specifically, it is looking to see which features of user model visualisations have proved successful, and difficulties experienced by users in understanding reflecting over their user models. It highlights the need for scrutable user models designed in line with Shneiderman's guidelines for powerful data visualisations [Kay, 99] [Shneiderman, 96].

With the explosion of the World Wide Web, the need for tools to support users in retrieving the information they require became apparent. Search engines have been developed to become better at finding answers to queries with minimal user input. Ambiguity in search queries prompted the development of Personalised Information Retrieval (PIR) systems which record searches, clicks, scrolling action and viewing behaviour on retrieved results to make assumptions on user interests. As well as these implicit feedback methods, some PIR systems explicitly ask for feedback from the user; however the move has been to implement as much implicit feedback as possible due to users' ineptitude at specifying their needs [Li, 11]. PIR systems are optimal for query searches where the user requires specific information on which the system zooms-in with successive iterations. Today, with the volume of knowledge available, a move towards doing research online has been observed. By honing in on a piece of information, PIR systems inhibit exploratory search.

With exploratory search, the information needs of the user are continuously evolving as the user gains more domain knowledge and changes the focus of their query. Users require answers to why, what and how questions rather than who, when and where questions typical in search queries. To cope with the limitations of search engines in exploratory search, users submit multiple queries to a variety of search systems in parallel, with multiple browser tabs open, selectively following links and recording and analysing any potentially useful information using the clipboard [Singer et al., 11]. Mendeley, a tool which assists users in the organisation of their collections of research documents, has become increasingly popular. Metadata can be extracted and text can be highlighted and annotated to support the exploration; however, there is limited support for personalisation [Holub et al., 14]. The use of hypertext when performing exploratory search allows users to link through different objects in a network structure; however as any course can be chosen, users may lose direction in their search. Recommendations made taking users' goals into consideration significantly accelerate the user's search for information [Kobsa, 94].

Personalisation in exploratory search has the added complexity of having to tailor the recommendations for the individual user based on their personal experience rather than a standard experience. The needs of a member of the general public will vary greatly from those of professional researchers due to the disparity in their prior knowledge of the subject. Differing levels of knowledge and experience may lead to suggestions that are too difficult and detailed for novices and too redundant for experts if the level of knowledge is not taken into account in the personalisation process [Kobsa, 94]. Furthermore, individuals tend to occupy a continuum of experience rather than be easily compartmentalised by level of expertise, and this continuum is expected to evolve as the user cultivates knowledge.

While recommendations need to cater for the users' knowledge level and experience, some applications may not require complete accuracy in the assessment of users' domain knowledge. In such cases, using the stereotype approach to model users has proven very useful [Kobsa, 93]. Grouping user knowledge levels as novice, intermediate and expert is a common example of this. The use of stereotypical assumptions over the accurate assessment of users' knowledge will increase the likelihood of inaccuracy. Kobsa suggests that users should be aware the system uses a user model, that users should be aware the user model may include interests which are not in line with those of the user, and that users should have the opportunity to view and possibly adjust the model but that allowing users to modify their model introduces the risk that users may “sabotage” their model.

Sweetnam et al. considered tailoring the interface to the individual users so that professional researchers would avail of a rich feature set while general public members were offered a simpler feature set [Sweetnam et al., 13]. However they found that access to the full feature set was required by all users for user development to be supported and enabled.

Exploratory search generally begins with an imprecise query and becomes an iterative process of exploring retrieved information and reformulating the query. It is difficult for technology to predict dynamically changing information needs that satisfy the constantly changing user knowledge. Much research has been undertaken using methods such as relevance feedback retrieval, faceted search and result clustering, to deduce from user actions whether suggestions are too broad or too narrow for the current information needs and to understand user behaviours in exploratory search information retrieval. However, these were found to be overly demanding on users. Instead it was found that interactive visualisations of the user model provided insight into the topic for users at the general public end of the continuum of experience and allowed users with more domain knowledge to provide feedback regarding their needs [Athukorala et al., 14].

Koenemann and Belkin undertook an experiment comparing three information retrieval systems [Koenemann and Belkin, 96]. In the first system, user model information was hidden from the user. In the second system, user model information was shown to the user but could not be modified. In the third system, the user could add and remove terms in the user model to better reflect their needs. It was found that the third system performed the best. User feedback indicated that they prefer having the ability to see and adjust the user model to better reflect their needs. The benefits of user interaction were also reflected in the reduction in the number of iterations required to complete the required task.

User models are generally represented visually as sets of weighted keywords or concepts [Gauch et al., 07]. While keyword user models are simplest to build, the volume of keywords can become difficult to manage and require excessive user feedback. Concept user models map vocabulary to concepts, making the model more manageable and requiring less feedback.

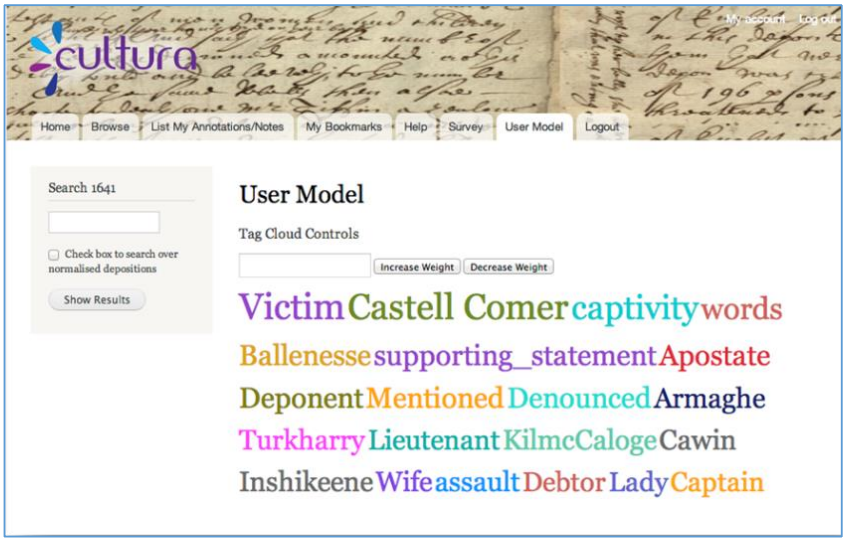
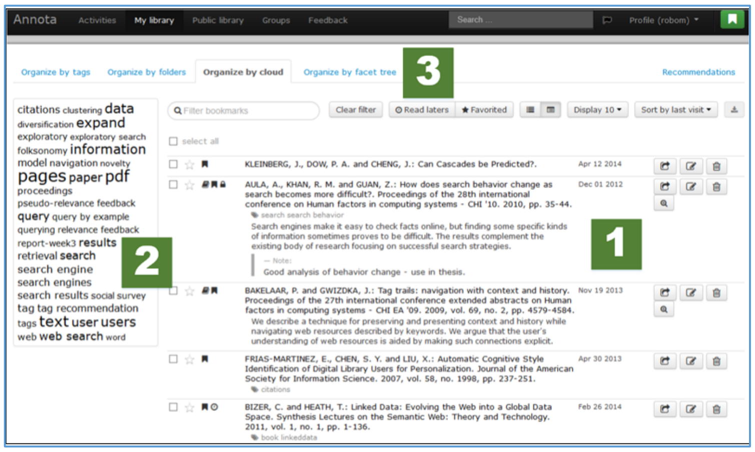

cultura
My account
Log out
Home
Browse
List My Annotations/Notes
My Bookmarks
Help
Survey
User Model
Logout
Search 1641
Check box to search over normalised depositions
Show Results
User Model
Tag Cloud Controls
Increase Weight
Decrease Weight
Victim Castell Comer captivity words
Ballenesse supporting_statement Apostate
Deponent Mentioned Denounced Armaghe
Turkharry Lieutenant KilmcCaloge Cawin
Inshikeene Wife assault Debtor Lady Captain

Figure 3.1: CULTURA Tag Cloud user model

A simple example of a user model visualisation is a tag cloud user model such as that used by the CULTURA Project, seen in Figure 3.1. As mentioned in Chapter 2, users can enter the reflect phase of the four stage personalisation approach at any time. This is where the user collaborates with the system to ensure the technology is providing the required support for that user. The user examines their user model to see what the system perceives to be their interests. They can then interact with the user model modifying weights or removing terms to give a user model that better reflects their actual interests [Hampson et al., 14].

The Annota system is similar to Mendeley in that it is a bookmarking service where text can be highlighted and annotated. However, it contains the added feature that the metadata is gathered and used to generate a user model which can be viewed alongside the collection of documents as a tag cloud, list of tags or a facet tree. Users can filter subsets of their personal collection through the user model visualisations. Figure 3.2 illustrates the Annota web interface with the user model in tag cloud format [Holub et al., 14].


Annota
Activities
My library
Public library
Groups
Feedback
Search ...
Profile (robom)
Organize by tags
Organize by folders
Organize by cloud
Organize by facet tree
3
Recommendations
citations clustering data diversification expand exploratory exploratory search folksonomy information model navigation novelty pages paper pdf proceedings pseudo-relevance feedback query query by example querying relevance feedback report-week3 results retrieval search search engine search engines search results social survey tag tag recommendation tags text user users web web search word
2
Filter bookmarks
Clear filter
Read laters
Favorited
Display 10
Sort by last visit
select all
1

KLEINBERG, J., DOW, P. A. and CHENG, J.: Can Cascades be Predicted?. Apr 12 2014

AULA, A., KHAN, R. M. and GUAN, Z.: How does search behavior change as search becomes more difficult?. Proceedings of the 28th international conference on Human factors in computing systems - CHI '10. 2010, pp. 35-44. Dec 01 2012
search search behavior
Search engines make it easy to check facts online, but finding some specific kinds of information sometimes proves to be difficult. The results complement the existing body of research focusing on successful search strategies.
— Note:
Good analysis of behavior change - use in thesis.

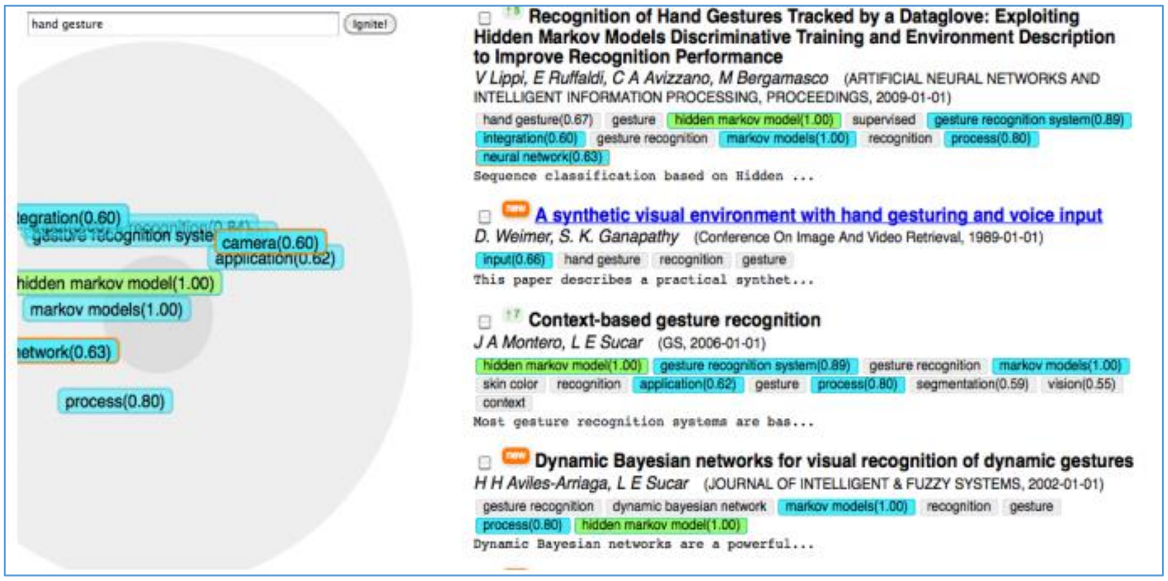
BAKELAAR, P. and GWIZDKA, J.: Tag trails: navigation with context and history. Proceedings of the 27th international conference extended abstracts on Human factors in computing systems - CHI EA '09. 2009, vol. 69, no. 2, pp. 4579-4584. Nov 19 2013
We describe a technique for preserving and presenting context and history while navigating web resources described by keywords. We argue that the user's understanding of web resources is aided by making such connections explicit.

FRIAS-MARTINEZ, E., CHEN, S. Y. and LIU, X.: Automatic Cognitive Style Identification of Digital Library Users for Personalization. Journal of the American Society for Information Science. 2007, vol. 58, no. 1998, pp. 237-251. Apr 30 2013
citations

BIZER, C. and HEATH, T.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology. 2011, vol. 1, no. 1, pp. 1-136. Feb 26 2014
book linkeddata

Figure 3.2: Annota web interface

The user interface of SciNet, a system developed to examine the concept of interactive user modelling, can be seen in Figure 3.3. SciNet is designed specifically to support exploratory search and cope with evolving user needs. The user model in SciNet is visible to the user at all times to the left of the recommended documents. Keywords are extracted from the user’s query and document list and users can increase or decrease their importance by moving them closer to or further from the centre. Ruotsalo et al. found that while SciNet requires users to take the time to analyse and modify the user model, it still outperforms Google Scholar in terms of finding relevant, novel and diverse information without extending the time required to perform tasks [Ruotsalo et al., 13].



Recognition of Hand Gestures Tracked by a Dataglove: Exploiting Hidden Markov Models Discriminative Training and Environment Description to Improve Recognition Performance
V Lippi, E Ruffaldi, C A Avizzano, M Bergamasco (ARTIFICIAL NEURAL NETWORKS AND INTELLIGENT INFORMATION PROCESSING, PROCEEDINGS, 2009-01-01)
hand gesture(0.67) gesture hidden markov model(1.00) supervised gesture recognition system(0.89) integration(0.60) gesture recognition markov models(1.00) recognition process(0.80) neural network(0.63)
`Sequence classification based on Hidden ...`

new **A synthetic visual environment with hand gesturing and voice input**
D. Weimer, S. K. Ganapathy (Conference On Image And Video Retrieval, 1989-01-01)
input(0.66) hand gesture recognition gesture
`This paper describes a practical synthet...`

Context-based gesture recognition
J A Montero, L E Sucar (GS, 2006-01-01)
hidden markov model(1.00) gesture recognition system(0.89) gesture recognition markov models(1.00) skin color recognition application(0.62) gesture process(0.80) segmentation(0.59) vision(0.55) context
`Most gesture recognition systems are bas...`

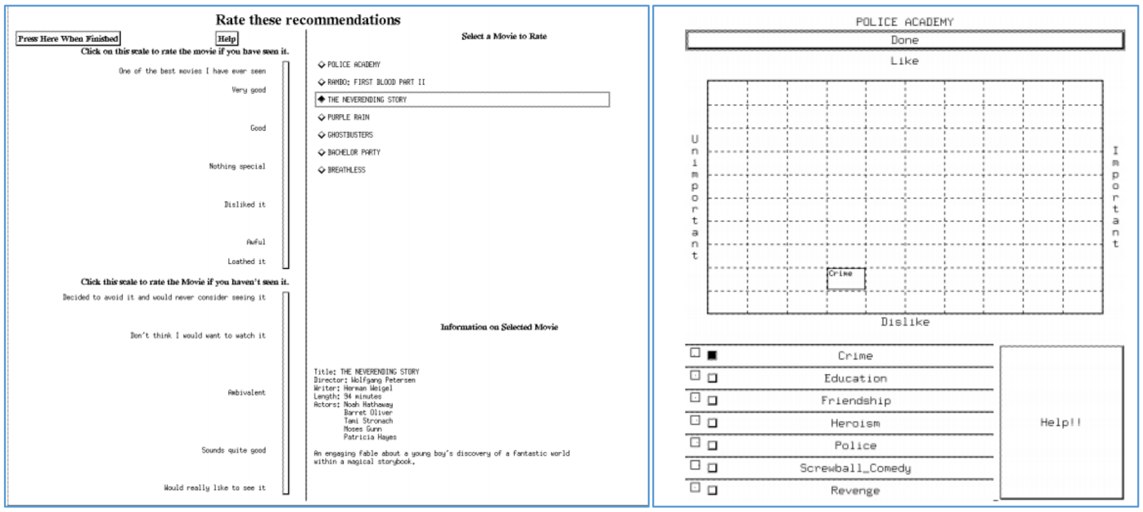
new **Dynamic Bayesian networks for visual recognition of dynamic gestures**
H H Aviles-Arriaga, L E Sucar (JOURNAL OF INTELLIGENT & FUZZY SYSTEMS, 2002-01-01)
gesture recognition dynamic bayesian network markov models(1.00) recognition gesture process(0.80) hidden markov model(1.00)
`Dynamic Bayesian networks are a powerful...`

Figure 3.3: SciNet user interface

The challenge with collaboration between the user and the technology is that it requires that the user be aware of their information needs, that they understand the information presented to them in the user model and that they understand the processes responsible for the generation of this model. Scrutable user models are models that can be examined or inspected closely and thoroughly. Kay describes some of the motivations for scrutability in user models [Kay, 99]. To generate the user model, the system must hold personal information about the user; users have the right to see what information is held about them and they have the ability to correct any errors in the model. Giving users access to their model also ensures programmer accountability in modelling the user's interests and portraying them in a way the user can understand. A critical argument for scrutability is that users can check and correct the model. Scrutability enables collaboration between the user and the technology each of which may interpret information encountered differently. The technology will have an objective approach, processing the information clinically which may not mirror the user's actual beliefs. Collaboration will ensure both parties have the same understanding of the user's interests. Finally, scrutable user models can be used as an aid to reflection. Through examining the user model, users can become more selfaware since the model reflects their real actions. This can help them monitor their progress and plan the next steps of their exploration.

In line with the requirements for a scrutable user model detailed above, Kay developed the UM toolkit, a user model shell designed to work for a range of different activities including personalised search and as a learning tool. It is predicated on the cooperation between the user and the system and relies on successful communication between both parties. Successful communication is achieved by the system building a user model, but different activities require different types of user models. The UM toolkit was tested on two systems – a coaching system and a movie advisor that tracks user preferences for different characteristics of movies. The movie advisor first models user preferences based on a user’s personal attributes and movie ratings submitted by the users from a list of movies to give a range of movie characteristics. Stereotype reasoning is applied to the personal attributes; for example, a teenage male would qualify for the stereotype set ‘teenage males tend to like action movies’. Preferences are inferred from user movies; for example, on registering a poor user rating on three science fiction movies, the system would infer that that user dislikes science fiction movies. The system may also infer information about the user that the user is unaware of; for example the system may identify a link between highly rated movies and a particular director which the user had not observed. The user model is then presented to the user to update before it starts to recommend movies based on these interests in the user model.

Once the initial user model has been set up, each time a user revisits the model they are asked to rate the previous recommendations. If the user dislikes one of the recommended movies, this is an indication that the user model is incorrect. The user is presented with a troubleshooter interface which they can interact with to adjust their user model. The recommender interface and troubleshooter interfaces are kept simple and intuitive but flexible, as can be seen in Figure 3.4. Through continuous communication between the user and the system, the system is able to recommend movies that coincide with the user’s preferences [Kay, 94].


Rate these recommendations
Press Here When Finished
Help
Click on this scale to rate the movie if you have seen it.
One of the best movies I have ever seen
Very good
Good
Nothing special
Disliked it
Awful
Loathed it
Click this scale to rate the Movie if you haven't seen it.
Decided to avoid it and would never consider seeing it
Don't think I would want to watch it
Ambivalent
Sounds quite good
Would really like to see it
Select a Movie to Rate
POLICE ACADEMY
RAMBO: FIRST BLOOD PART II
THE NEVERENDING STORY
PURPLE RAIN
GHOSTBUSTERS
BACHELOR PARTY
BREATHLESS
Information on Selected Movie
POLICE ACADEMY
Done
Like
Unimportant
Important
Crime
Dislike
Crime
Education
Friendship
Heroism
Police
Screwball_Comedy
Revenge
Help!!

```
Title: THE NEVERENDING STORY
Director: Wolfgang Petersen
Writer: Herman Weigel
Length: 94 minutes
Actors: Noah Hathaway
        Barret Oliver
        Tami Stronach
        Moses Gunn
        Patricia Hayes

An engaging fable about a young boy's discovery of a fantastic world
within a magical storybook.
```

Figure 3.4: UM toolkit movie advisor recommender interface (left) and troubleshooter interface (right).

Visualisation methods and techniques have been evolving rapidly to deal with the colossal amounts of data that can be collected as a result of the web and emerging technologies. The study of the human eye has revealed that certain features of visual objects are perceived before the eye has time to react. These include size, shape, colour and direction of motion. These features can be exploited in visualisation to highlight information about the node, or its relationship to other nodes. A change in the size of a node will correspond to a change in value and relative sizes will give users an indication of relative importance. Colour scales are used to depict a scale from one extreme to another which can be understood by users very intuitively. Peaks and troughs in a timeline graph will attract the user's attention before uniformly shaped regions. Motion, even in the periphery, will attract the user's attention and is intuitively indicative of the change in the data. These features have therefore been built into computerised visualisations [Diehl, 07].

As the number of dimensions in data elements collected has grown, two approaches have been developed to support the analysis of large multi-dimensional data sets – multi-dimensional visualisations and multi-view visualisations. Multi-dimensional visualisations incorporate all of the dimensions in one visualisation. These were found to be difficult to generalise for all data sets and for users to comprehend and navigate intuitively [Shneiderman, 96]. Instead there has been a shift to multi-view visualisations where different dimensions of the data set are displayed side by side in a number of views.

Shneiderman examined various visualisation designs and summarized the basic principle into what he named the Visual Information Seeking Mantra [Shneiderman, 96]:

"Overview first, zoom and filter, then details-on-demand"

That is, users should be given an overview of the entire data first. They should then be able to zoom into items of interest, filtering out any items not of interest. Finally they should be able be presented with details on remaining data as required. To transform data exploration a more "joyous experience" Shneiderman proposed 3 further steps – relate, history, extract. Relate refers to the relationship between attributes. Shneiderman proposed that once the user has selected a subset of a particular attribute, they should be able to observe any effect on other attributes of the dataset caused by the reduction to that subset. This could be achieved through linking visualisations so that if the user applies a filter to the data in one visualisation, the change in the dataset is propagated through the rest of the visualisations. History refers to the ability to keep of track user interactions so as to replay or refine them. Extract refers to the ability to save a particular state of interest to share with others. Shneiderman suggested that these features would greatly assist in the exploration and analysis of large data sets.

The CULTURA user model provides a certain degree of scrutability. Users can see the information held by the system and can collaborate with the system to adjust their model to best reflect what they perceive to be their needs. The model, however, does not present users with sufficient information about the generation of the user models to facilitate self-awareness. Users cannot see a temporal span where entities become prominent and drift away as the user steps through the digital collection. As such they may question the validity of the system-generated user model due to not understanding the rationale behind the presence or importance of certain entities in the model. An overview of the current user model is provided, but none of the other steps proposed by Shneiderman are provided by this model.

In terms of scrutability, the Annota web interface presents the user with the information it holds about them, but there is no facility to correct errors or collaborate with the system. An overview of the user model is provided and can be used to zoom in and filter the collection with documents in the subset displayed on the right of the screen. However, it does not cater for the relate, history or extract steps proposed by Shneiderman.

SciNet provides a scrutable user model. The information held by the model is displayed to the user at all times. The user is encouraged to scrutinise the model and collaborate with the system to best meet their needs. Through continuous collaboration, the user should be aware of the processes that have led to the generation of the current model. In terms of the Shneiderman steps, however, it only provides an overview of the user model. In particular, the fact that there is no facility to replay the evolution of the user model may lead to issues where a user performs the exploration over a long period of time. As a result, it may restrict their ability to reflect over their progress.

The UM toolkit movie advisor adheres to the scrutable user model guidelines and has been found to make pertinent recommendations; however it requires continuous user feedback which users may find obstructive during exploratory search. The system can deal with changes in the user's preferences; however it does not provide a facility to reflect over how their preferences have changed over time. While this feature would not be useful for movie recommendations, during exploratory search, the ability to reflect over the path taken during the exploration can help users understand how they have reached their current point and reorient their exploration as required. In terms of Shneiderman's steps, an overview of the user model is provided but these only show the film categories. There is no facility to zoom into other characteristics on which the system bases its recommendations, such as directors or actors. The user can filter on the information provided; however this does not provide further insight into the data presented. Details-on-demand are available in the form of the labels on the troubleshooter interface. The relation between highly rated movies and directors could be of interest to the user; however, this system does not cater for the relate step, nor does it cater for the history or extract steps.

This chapter has compared the requirements for query search and exploratory search, discussed some the challenges in developing effective user models and considered some of the current frameworks that support exploratory search. The requirements for a scrutable model were outlined and a scrutable recommender system was detailed. Desirable features of visualisation designed were outlined and the frameworks considered in this Chapter were evaluated against these features and the requirements for a scrutable model. The next chapter looks at technologies that will be used to develop a framework to generate scrutable and controllable user models to support reflection through a coordinated visualisation satisfying the first six of Shneiderman's steps.

4 Related Technologies

In order to develop a framework to generate scrutable and controllable user models to support reflection through a coordinated visualisation, a number of technologies are required to collaborate with each other. The framework was developed using a Client-Stateless-server architecture. This enables the separation of the user interface and the storage components of the system, keeping all of the session state on the client side, not on the server side. This chapter begins by looking at the technologies employed on the client side. The client side is split into baseline technologies common to most web applications, visualisation technologies used to create the coordinated visualisation system for this project, and communication technologies from the client side used to communicate with the server. The server side is then examined, outlining what is meant by a "stateless-server" and how this is achieved, and detailing the technologies used to develop and test the server-side and store the various sets of data.

4.1 Client-side

4.1.1 Baseline Technology

HyperText Markup Language (HTML) is used to create the user interface. HTML is one of the most commonly used languages for the creation of web pages. HTML elements are used to build the structure and content of the interface. Element attributes dictate their type, function and style. An identification tag can be assigned to an element so as to assign methods to that element or monitor interaction with the element. Elements may also be declared as part of a class to apply formatting or assign methods to a group of elements and to monitor interaction with the group of elements. The framework creates a group of carefully selected elements to facilitate the development of a coordinated visualisation system which will yield insight into the user data. Additional elements are included around the group of coordinated visualisation elements to provide various methods for the user to select historic events and provide additional information beyond the coordinated visualisation system [HTML].

The formatting and layout of a webpage can be done within the HTML document, or the HTML document can be linked to one or more Cascading Style Sheets (CSS). In this framework, a number of CSS plugins are used. Bootstrap.css is used to format the webpage. jQuery-ui.css is used to format hover boxes which provide additional information for the user. DC.css is a CSS file which accompanies one of the visualisation libraries which will be discussed in the next section. In the CSS, the size of elements, their location relative to each other and their location within the page can be set out [CSS].

HTML provides the building blocks for the user interface and CSS styles it. But the real work happens in the JavaScript files. Once the web page has been loaded and the elements have been created, any JavaScript file linked to from the HTML document is compiled. JavaScript handles the background activity of elements defined in the HTML file, such as the customisation and manipulation of the elements or computations related to the elements. Values and functions can be added to, and actions can be triggered on interaction with, HTML elements using JavaScript [ECMA]. In this project, a JavaScript file is used to tie together all of the client-side operations. It links the HTML and CSS files to the JavaScript APIs, jQuery event triggers and AJAX requests.

4.1.2 Visualisation Libraries

Three JavaScript APIs are employed in the coordinated visualisation system used in this project:

- D3.js
- Crossfilter.js
- DC.js

The Data Driven Documents JavaScript library (d3.js) combines HTML, CSS and Scaled Vector Graphics (SVG) to bring data to life [D3]. Countless different types of visualisations can be produced using d3.js which can, if required, be animated and interacted with. SVG objects are created within a HTML document and styled through CSS initially. Using the d3 library, the charts can then be set up so that users can interact with the visualisations. For example, on selection of nodes, the formatting can be set to change or if the node is a parent in a hierarchy, users may delve deeper into an element. Visualisations may be animated through the use of transitions to apply changes gradually. Through binding the data to the SVG, features of the visualisation may be used to represent the data; for example, the data may be bound to a bubble chart in such a way that the radius of the nodes are determined by one of the attributes of the data set. D3.js is a powerful library in the sense that it can produce an extensive variety of creative visualisation; however, it requires that the data be saved in specific formats in order to filter down into a particular Dimension. Data therefore must be manually formatted to suit different representations in a coordinated visualisation. Applying Filters to one visualisation would only be applied to the data set for that visualisation; changes would not be propagated through to other visualisations in the group. To overcome this issue, the Crossfilter API is used.

The Crossfilter JavaScript Library facilitates high-speed multi-dimensional filtering, even for datasets containing in excess of 1 million records [Crossfilter]. The Crossfilter is the term given to the full multi-dimensional dataset, usually made up of an array of JavaScript objects. In this framework, the data will be retrieved from the server in JSON format. The Dimension specifies the aspect of the Crossfilter data the user is interested in. The aspect may be an attribute of the dataset, such as location, or it may be derived from one or more attributes, such as the year derived from the data, or profit derived from purchase and sale prices. The Filters that can be applied to a Dimension are what the Dimension can be split into. For example, if the Dimension is year and the years range over the 1990s, the Filters will be 1990, 1991,…, 1999. The user could look at the full decade, at individual years or at a selection of years, such as {1994, 1995, 1996} or any combination of the 10 years, simultaneously. The number of records in each Dimension with Filters applied is determined by grouping the Dimensions and returning the number of elements present to be used in the rendering of the visualisation. The issue of formatting the data sets for each visualisation when using the D3 library is overcome using the Crossfilter library in the way the Dimensions are defined.

The Dimensional Charting JavaScript Library (dc.js) combines the D3 and Crossfilter libraries to build coordinated visualisations [DC]. Chart objects are created to link the HTML elements representing the charts to the DC chart functions. Various properties and functions are assigned to the charts, dependant on the type of DC chart concerned. When Filters are applied, different charts are handled differently, depending on whether an array of Filters or a range is applied and which features need to be recalculated, such as the radius of nodes or height or width of bars. The charts are created within a chart group so that any interaction with one of the charts in that group will be propagated onto the other charts within that group. A Crossfilter is constructed with the full dataset and the required Dimensions for the various graphs are defined. The D3 chart attributes are defined as required using the relevant Dimensions or values derived from the Dimensions as data input. When a Filter is applied to a chart, the DC API calls on the Crossfilter API to recalculate the number of elements in each remaining Filters of each Dimension. It then calls on the relevant D3 methods for each chart to recalculate the data bound parameters of the charts and redraw all of the charts. Therefore, as a user interacts with the visualisations, any Filter applied to one visualisation is propagated through to all of the other visualisations within that group.

4.1.3 Communication

User log data must be processed to determine the interests and their related weights at each interaction which feed into the user model. This involves the retrieval of data from various sources and the storing of data. If the data has been processed and stored, the user model data must be retrieved from the server. During interactions with the coordinated visualisation where the user changes the point in time at which they wish to view the user model, additional information about the interactions must be retrieved from the server. To store and retrieve data, two more technologies are required on the client-side – jQuery and AJAX. As the data is pre-processed, a number of sequential functions are executed, some of which contain AJAX requests to send and retrieve data to and from the server. As the user views the user model at different interactions, jQuery event handlers are triggered which contain AJAX requests to retrieve data from the server.

jQuery is a JavaScript library that facilitates event handling [jQuery]. As the user interacts with different HTML elements on the web page, event handlers are triggered corresponding to specific interactions with a specific HTML element or group of HTML elements. This can be done by element id or by class to which a group of elements belong. For example, changing the event using the slider will require an individual event handler, while a group of charts belonging to the same coordinated visualisation system will require an event handler for elements in that group or class. Where the event handler is defined for a class of elements, the id of the element with which the interaction occurred can be extracted using jQuery functions to keep track of changes made to the filters.

Asynchronous JavaScript and XML (AJAX) requests are a part of the jQuery API [AJAX]. They allow HTTP requests to be performed asynchronously. The use of AJAX to perform HTTP requests allows relevant features to be updated when data is retrieved from the server rather than reloading the page. The HTTP method and URI of the relevant resource are provided in the AJAX request to the server. The most commonly used HTTP methods [HTTP] are:

- GET to retrieve data from the server
- PUT to update data on the server
- POST to send data to the server
- DELETE to delete data from the server

While the name suggests the sending and retrieving of Extensible Markup Language (XML) data, JavaScript Object Notation (JSON) is more commonly used and will be used in this framework.

JSON are made up of key/value pairs [JSON]. Before sending data to the server, the data object made up of a collection of key and value pairs must be converted to a JSON string using the “stringify” function. On retrieving data from the server, the JSON string must be converted to the data object made up of a collection of key and value pairs using the “parse” function.

4.2 Server-side

PHP Hypertext Preprocessor (PHP) is a scripting language used for code which is executed on the server [PHP]. To support the development and testing of web applications before uploading them to the actual server, test servers are commonly used to execute the PHP code. AMP servers facilitate the creation of database driven applications using an **A**pache HTTP server, a **M**ySQL database and **P**HP. This framework was developed using WAMP, an AMP server on **W**indows [WAMP].

Chapter 2 outlined the technologies used by CULTURA to log user interactions and extract entities and their relationships from the resources. The log data was provided in the form of an XML document. XML documents consist of content wrapped in tags; PHP SimpleXML facilitates the reading of XML data from a file, outputting the data in the form of an object. The object can then be traversed using PHP script to extract the required information.

The client communicates with the server via a Representational State Transfer (REST) API. REST APIs are stateless; that is, no session state is saved on the server [Fielding, 00]. The client must provide any data required to perform a request. If the client requests the retrieval of data, the data must be retrieved from some storage space, in this case a database. This allows each request to be dealt with independently, partial failures to be dealt with without terminating the session and multiple clients to operate with the same server.

The REST API used for this framework is the Slim Framework. The Slim Application is defined in a PHP file where combinations of a resource and a method provide the route to the appropriate anonymous function to be executed. A .htaccess file is used to reroute the URI from an AJAX request to this PHP file in order to get the required Slim Application route. Each AJAX request contains a URI composed of the location of the PHP file with the resource name appended to it. The HTTP method describes what is to be done with the resources. The Slim Application uses the part of the URI identifying the resources and the HTTP method from the AJAX request to determine the route to the required anonymous function.

The anonymous functions within the various Slim application routes make up the server side of the framework. Due to the stateless property of the REST API, no data is saved on the server. Data received from the client must be decoded before server actions, such as querying a database to store or retrieve data, can be performed. Data must be encoded before it is returned to the client’s callback function which instantiated the AJAX request.

For added security, PHP Data Objects (PDO) are used to access the database. Prepared statements and data objects replace standard SQL queries. When data from the client is received by the server, the prepared statement is executed with the relevant data in the required placeholders. This protects against SQL injection attacks. With PDO the format of the query results can be specified. For example, FETCH_OBJ returns records as objects using the database attribute names as the properties of the object, while FETCH_ASSOC returns records as arrays where the array index represents the column number.

Through using databases, sets of data can be stored in different tables and retrieved in later sessions. The semantic data for the CULTURA collections was procured in the form of MySQL data dumps. MySQL has the facility to easily import data dumps from other databases, creating and updating tables as required to replicate the database from which the data comes. Databases may be queried to retrieve subsets of the data, to update records and to append to sets of records interchangeably. Queries are performed quickly so that actions appear instantaneous to users. MySQL is currently the most popular open source database which is used for web applications. One reason for this is that it was designed and developed specifically for web applications. It is used by companies such as Google and Facebook for storage and fast retrieval of large volumes of data [Oracle, 13].

This chapter gave a brief overview of the various technologies used in this project to develop a framework to generate scrutable and controllable user models to support reflection through a coordinated visualisation. Further insight into how these technologies interact and what features in particular are exploited to realise the framework will be provided in the Design and Implementation chapters where required.

5 Design

Digital collections are a great source of knowledge to which access can easily be gained; however trawling through voluminous corpuses for information of interest can be a laborious, time-consuming task. Tools that collaborate with users to model their interests and subsequently use this model to narrow the collection under consideration have been found to expedite this task. To enable effective collaboration, users must understand their needs and how they have evolved throughout their exploration. Visualisations of the user model held by the system can assist users in their reflection over their interests; however static snapshots of user models may not provide sufficient information for users to extract valuable insight. Scrutable and controllable user models should be presented to users to support their reflection, enabling fruitful collaboration between the user and the system. This project looks at developing such a framework using the technologies introduced in Chapter 4. In this chapter, the process undertaken to design this framework is outlined. The goals and objectives which the design must meet are first examined. The data required to generate user models is identified. The coordinated visualisation design and the motivation behind the design are detailed and data requirements to achieve this are studied. The data storage design is explained. Finally, the overall design of the system is outlined.

5.1 Goals and Objectives

In order to generate scrutable and controllable user models over which the user could reflect on the evolution of their interests, the key features that would support reflection were identified.

There is a balancing exercise in the designing of user models and that is finding the balance between not providing sufficient information, whereby the user will not extract useful meaning, and providing too much information, which runs the risk of overwhelming the user. Different users may also desire different levels of information, depending on the type of exploration being undertaken and the experience of the researcher. It was therefore decided that users should be able to throttle the amount of data displayed in the model.

Having zoomed into the required level of data, users may still find that there is insufficient information provided surrounding the rationale for the presence and relative importance of interests displayed. A need for a means of representing the '*why'* was required; that is why interests were present and why they were assigned a particular importance relative to other interests.

During the user-system collaboration, users need to gain a good understanding of how their interests have evolved and what their current interests are so that they can feed back into the system. To facilitate this they may require the ability to look at subsets of the data having particular properties, such as people or locations.

Due to the complexity of the data involved to meet these user requirements, it was decided that the user model should be designed using coordinated visualisations to support the exploration and yielding of deeper insight of the data. With these user requirements in mind, the following goals were set:

- Store sufficient information about the user model at each event to be able to recreate all historical user models as well as the current user model.
- Present a user interface that is easily adopted by users to support reflection.
- Render the user model for any event highlighting changes compared to the previous event and providing useful human-readable details on demand of the changes.
- Facilitate the performance of complex analyses on the user model.
- Provide tools to view useful subsets of the model to gain a deeper insight into the user model.
- Allow the user to throttle the level of information presented to focus on primary interests only.

- Allow the user to throttle the level of information presented to analyse secondary interest so as to identify changes in the focus of their exploration and possible serendipitous relationships.
- Enable users to reflect on how the model, and hence their interests, have changed over time.

For these goals to be met, the following objectives must be realised:

- Analyse and understand the log data so as to identify the structure of the logs of the different user actions, the information required to create the user model and the most general method of extracting the required information from the log data.
- Analyse and understand the semantic data for the collection of documents on which the user actions are performed so as to identify how the log data can be used to identify the related entities and the minimal entity information required to create the user model and provide users with human-readable descriptions.
- Design the coordinate visualisation interface, identifying feasible filtering tools that will be most beneficial for users to gain insight into their user model, thus supporting reflection.
- Design the shell of the system, identifying what technologies are needed and how they should communicate.
- Design the inner working of the system, what should happen during the pre-processing of the data, during the initial rendering of the visualisation and during the user's reflection of the progress of their exploration.
- Implement the technology to meet the required goals in line with the chosen designs.
- Design an appropriate task based evaluation and test that the technology works as expected and meets the required goals.

5.2 User Model Data Requirements

The first step in the design of the framework was identifying the data that would be required to realise the objectives and generate a user model that achieved the goals outlined. In this project, the user model data was generated from the semantic and log analysis of data acquired from the CULTURA project.

The primary features that make up any user model were identified to be the interests and their associated weights. These are aggregated over time so that each user model feeds into the next user model.

In order to determine the interests of the user, the log data was analysed to identify what information could be extracted from users' actions that would provide insight into their interests at each interaction with the system. It was identified that the log data contained resource identifiers, indicating the resource they had interacted with. These identifiers corresponded to unique elements in the semantic data and could therefore be used to link the log data to the semantic data.

An analysis of the semantic data found that the resource identifier would represent either documents or entities; document identifiers could be used to extract the entities present in the document; entity identifiers could be used to extract entities related to that entity. The entities extracted using the document and entity identifiers would be used as the interests at each interaction.

Weights needed to be assigned to the interests present in the user model to represent the relative importance of interests. User actions can be indicative of the importance of entities. For example if a user browses haphazardly, they are demonstrating some level interest in the entities they encounter through following links that catch their attention; however these entities may not be of particular importance to the user, they may merely be entities they encounter briefly. If however a user bookmarks or annotates a passage in a deposition, this illustrates that the resource is of particular importance to the user. Using low weights for page visits allows entities that are not particularly important to users to drift into and back out of the user model momentarily without impacting the recommendations made by the system. Interests will only begin to impact the recommendations if they are encountered repeatedly. Entities encountered in a bookmarked document can be assigned a heavier weight as the user has explicitly demonstrated an interest in that document; an interest in the entities within the document can therefore be inferred.

Through analysing the user log data, the various interactions were identified. It was observed that some interactions did not contain corresponding resource identifiers; however, for most this was not problematic as this was due to the fact that users had not interacted with a particular resource. It was decided that the text search would not contribute to the user model. The rationale for this was that text searches were ambiguous and often found to be misspelled. Identifying every entity containing the text and identifying the related entities for each of these entities would introduce many inaccuracies into the model. Instead it was reasoned that if the text search returned a resource of interest, the user would interact with that resource; that interaction would be logged and incorporated into the user model.

Entities used in entity searches could not be incorporated into the model as the log data did not contain the resource identifier. Similarly, the entities present in annotated text were not distinguishable; only the identifier for the document in which an annotation was made was provided.

The interactions that were considered of interest were annotations, bookmarks, visualisations and pagevisits. Pagevisits were seen to occur most frequently and were assigned the lowest weight since, as previously described, users would encounter many entities which did not hold much importance to them while browsing. Visualisations were ranked above pagevisits as they indicated an interest in a particular resource and the related entities. Bookmarks were ranked above visualisations as bookmarks indicated a desire to return to the document; hence signifying entities in the document were of particular interest. Finally, annotations were ranked above bookmarks as annotations were taken to indicate a particularly important piece of information.

As a user's interests evolve, some interests in their user model become more prominent as others drift away. Becoming more prominent indicates that the interest has a continued relevance to the user's exploration and the weights of successive interactions accumulate. If, however, the orientation of the user's interest deviates, entities that are no longer of significance to the user model must be phased out. To achieve this, at each interaction, entities that are not related to the current resource but are in the user model must be decayed. This should be done gradually to ensure prominent interests are not immediately phased out. This reflects the fact that generally during exploratory search researchers will gradually change the course of their exploration. Two methods were considered – a proportionate decay and a constant decay. A proportionate decay would penalise high importance interests more heavily while low importance interests would remain approximately stagnant, never exiting the model; therefore a constant decay was chosen for the design. The possibility of decaying based on time between interactions was also considered; however exploratory search can be carried out over long periods of time with clusters of activity separated by periods of inactivity during which interests remain constant. It was therefore decided that entities would be decayed per interaction, independent of the time between interactions.

Having analysed the log and semantic data as outlined above, it was concluded that:

- Log data for a particular user would need to be extracted from the full set of log data.
- The action and resources interacted with would need to be extracted from the log data for each interaction.
- The resource would be used to identify the entities of interest at that interaction.

- The entities of interest would be assigned a weight and added to the existing user model either as a new interest if it was not already present, or by adding the weight to the current weight for that entity.

5.3 The Coordinated Visualisation System

A simple user model can be generated using the interests and corresponding weights; however it will provide little insight into the '*why'*. The model should be scrutable; users must be able to see and understand the information held about them by the system, and in particular understand why any discrepancies between the user model presented by the system and their own beliefs of their interests exist. The model should also be controllable, adhering to Shneiderman's guidelines – *overview first, zoom and filter, details-on-demand*, *relate, history, extract* – where possible.

5.3.1 Visualisations

In accordance with the goals of this project, the user interface must be easily adopted and must facilitate the performance of complex analysis so that users may gain a deeper insight into the data. Coordinated visualisations assist in the exploration and analysis of large data sets. Through using coordinated visualisations, the full data set can be presented, meeting the scrutability criteria that data held by the system should be presented to the user. This is in line with Shneiderman's *Overview First* guideline.

Visualisations can contain information beyond the axis parameters through the use of colour, size and animations. The main features that will be of interest to users will be their interests and the importance of the interests. It was decided that a bubble chart would be used to illustrate the user model. Nodes would display the human readable interest name and their radii would be set using their importance. This would allow users to easily identify the most important interests at first glance. Beyond the interest and its importance, other features that were deemed would be of interest to users were whether the interest had increased or decreased in importance in the last event and how often it had contributed to the user model. The former would allow the user to differentiate between entities related to the resource interacted with in the event being viewed and those that had decreased in importance; this was represented in two ways. A colour spectrum was used with green representing an increase in importance and red representing a decrease in importance. The y-axis of the user model was used to represent the percentage increase in importance ranging from -100% to +100%. Interests which are new to the model will appear towards the +100% level; interests which are exiting the model will appear towards the -100% level. More persistent interests will appear around the 0% level. The number of times the entity has contributed to the model was represented using the x-axis. This would allow users to differentiate between entities of similar importance that appeared a different number of times. For example, supposing entity A appears 5 times in an annotated resource while entity B appears 20 times during page visits. This may result in the entities having similar weights but the user may deem entity A more important and wonder why entity B is assigned the same importance. By illustrating the disparity in the number of frequencies, the user will gain a better understanding of how the system has determined the importances.

Presenting the full user model including all entities of little importance can be overwhelming and too complex for the user to understand or check the validity of the data. Coordinated visualisations support the throttling of the data set to zoom into a particular tier of the dataset, and the filtering of the dataset to view subsets of the data based on particular attributes of the data.

In terms of throttling the data, users may be interested in viewing the full set of data to see what entities are entering the model or imminently leaving the user model; however the more important interests will be the ones contributing to the recommendations made by the system. As such, they may wish to be presented with interests above a certain threshold, the level of which may vary for different users. Interests believed by users to be in their set of primary interests should be present in the top tier of the user model dataset. Limiting the view to the top tier interests will allow users to check the validity of the user model, another of criteria for scrutable models. One of the benefits of reflection is the potential for serendipitous discoveries. Users may make some serendipitous discoveries in the top tier; however the secondary interests may hold more insightful information. Secondary interests will contain interests whose importance is on the decline but, up until recently, had been dominant interests. They will also contain rising interests which the user may already believe to be in their primary set. Zooming in on the data in terms of importance relative to the most important interest was therefore identified as a beneficial feature to promote scrutability. To provide flexibility in the upper and lower limits of the data under observation, a visualisation in which the user can select the required range was deemed most appropriate.

The semantic data was analysed to determine which properties the user may want to examine in isolation. Entities in the CULTURA 1641 collection were found to be classified into categories based on whether they were a person, place or crime. This was identified as a property users may wish to filter the data on. For example, they may be interested in examining the persons in isolation in hope of making a serendipitous discovery about relationships between persons. The use of pie charts to filter on the categories has the added benefit of displaying the proportion of the data that belongs to each category. This could identify, for example, that a user’s interests are heavily weighted by crimes.

While the entities that appeared in the event under examination will be coloured green and appear above the 0% line, users may also wish to only view these entities in isolation in the user model; or they may wish to examine which of their most important interests had not appeared. The ability to filter on increase or decrease in importance was therefore incorporated into the design to provide deeper insight. Similarly to the categories filter, pie charts work well for such a filter as users can approximate at a glance the proportion of entities in the model that have increased and decreased.

As per Shneiderman's guidelines, details should also be provided on demand. Users must also be presented with human-readable information, but too much text will clutter the user interface. It was decided that the human readable format of the entity names must be provided and used to label the users' interest; however additional information about changes in importance or the category the entity belongs to should only be displayed on demand. This would be displayed when a user hovered over a node. Additionally, the data presented in the user model would be listed in the form of a table below the user model. This table would contain the human readable name of the interest, the category to which it belongs, the previous weight, the current weight, the change in weight, the number of times it contributed to the user model and the number of events since it was seen. Providing this information adds transparency to the user model and equips the user with sufficient information to gain valuable insight.

A main feature required to achieve the goals set out was to allow users to cycle through past events to view and reflect over how and why their user model had evolved through the exploration. To facilitate the selection of events, two methods were chosen to be implemented – the use of a slider and the use of a text box. Details would be displayed about the event under review above the user model, identifying the resource, the type of action and date and time when the interaction occurred.

Bringing everything together, the user interface would be comprised of

- A bubble chart to represent the user model with additional details displayed when hovering over a node.
- A slider and a text box to select the event.
- A range chart to select the range of importance relative to the most important interest.
- Pie charts to filter on categories.
- A table listing all data currently in the user model, listed by importance.

By including these features in the design, users would be presented an **overview first**. They could then **zoom** into the tier of the dataset of interest and **filter** on categories and Increase/Decrease as required. Users could view **details-on-demand** by hovering over the nodes or examining the table below the user model. With filters applied, they could move through the **history** of their user model, allowing them to **relate** entities by observing concurrent or opposing trends.

A common issue with coordinated visualisations is the trade-off between the size of the data representations and the ability to view all visualisations at once on the screen. The layout of the interface was designed with the user model located between the filter tools and the data table. This would allow users to view the user model with the zooming and filtering tools and the first few table entries. Having applied the required zoom and filter, users could scroll as required to view additional table entries while still viewing the user model. Using the filtering tools while examining the data table was deemed less probable than using the filtering tools with the user model or than viewing the user model with the table of data entries for additional details-on-demand.

5.3.2 Data Requirements

Having determined the design for the user interface, the minimum data required to produce the visualisations and provide the additional detail was determined to be:

- The event number to identify the user model each set of data belongs to.
- The resource, action and date for the interaction
- The human readable name of the interests present in the user model at each event and their associated weights
- The previous weights of the entities in a user model to determine whether they had increased or decreased.
- The frequency of occurrence.
- The category to which the entity belongs.

All additional data required for the visualisations would be derived from this data.

5.4 The Databases

This framework utilises three data storage units. The first stores the user log data. An XML document is used in this project; however a database could be used instead. Log data for an individual user is extracted from the full log data set by the server and sent to the client requesting the information.

The second data storage unit is a database containing the graph links of entities and relationships between entities in the CULTURA collections. Using the resource identifications from the log data for each interaction, entities related to the resource are extracted with their human readable name and the category to which they belong.

The client uses the entity data returned with the action type of the interaction to determine the weight or increase in weight of the entities related to the current resources and degrades the weight of entities not related to the current resources. Data for each event is stored in the third storage unit, also a database. For each interaction by a particular user, the event number, date of interaction, type of action and entity description are stored in the events table. For each entity in the user model the event id, entity id, previous weight, current weight, frequency and number of events since the entity contributed to the model are saved in a second table. A third table relates the entity id to the entity details by storing the id with the description and category of the entity. The separation of event details and entity details from the user model data reduces duplication of data. An additional table also keeps track of the users whose data has been processed.

5.5 Linking the Technologies

Having satisfied the user model data processing and coordinated visualisation design objectives, the next objectives were to design the shell and inner workings of the system.

Figure 5.1 illustrates the architecture of the framework. The client or browser communicates with the server via a REST API. The server stores data received from the client into the required database, retrieves data requested by the client from the databases and sends it to the client via the REST API.

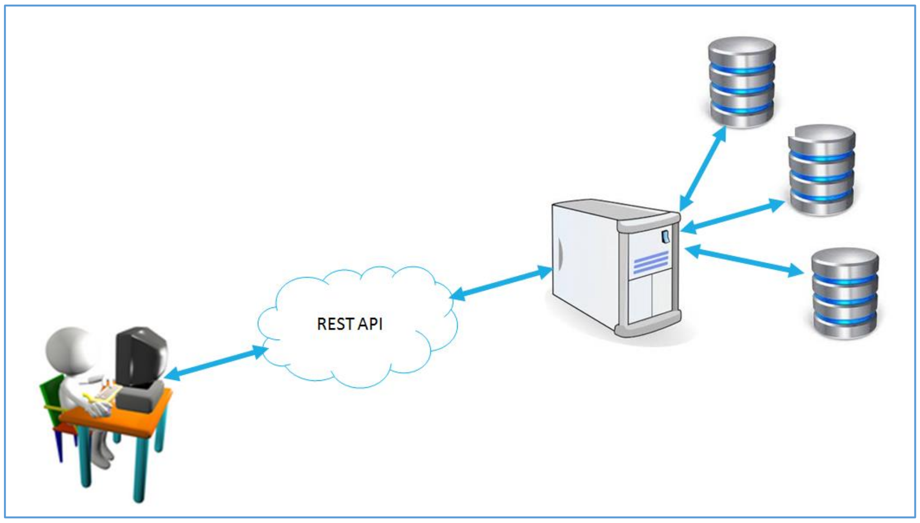


Figure 5.1: Overall architecture of the framework

The technologies used by the client side and the server side were detailed in Chapter 3. The final design of the system was as follows:

- The coordinated visualisation system would be created using HTML, CSS and the dc.js library which incorporates the D3.js and Crossfilter.js libraries.
- jQuery would be used to trigger event handlers when the user interacts with the interface.
- Where event handlers require the sending or retrieval of data from the database, AJAX requests are used on the client side.
- The Slim Framework would reroute the AJAX requests to the anonymous function required to deal with that request.
- The anonymous functions would make up the server in the framework. They would query the database to create tables, insert records, update records and retrieve partial or full records. Where required, they would return data to the callback function from which the request originated on the client side.

- On receiving data from the server, the client would perform the required actions with that data, for example calculating the weights of interests in the user model.

In terms of the inner workings of the system, the system should first check whether the user data had already been processed. In the case where it had not:

- The client should request the data from the server; the server should extract the log data for that user from the XML document and send the retrieved data to the client.
- The client should process each interaction, using the resource to request the related entities from the server. The server should retrieve the information and return it to the client.
- The client should determine the user model data that needs to be stored at each interaction and send the data to the server. The server should store the data.
- Once all of the data had been processed, the coordinated visualisation should be generated for the most recent event.

In the case where the user had already been pre-processed, the client should request the data for that user; the server should retrieve it from the user model data base and return it to the client. The coordinated visualisation model should then be generated for the most recent event.

As the user interacts with the visualisation, the DC API takes the filter from the Crossfilter API and calls on relevant D3 methods to recalculate the data bound parameters of the charts and redraw the charts. When the event is changed, the dataset should be restricted to data from that event and the charts should be redrawn. The client should request the event information from the server, the server should then query the user model database and return the required data. This data should be used to update the event details on the interface.

While this chapter concentrated on the design side of the framework, the next chapter will look at how this was implemented.

6 Implementation

Having identified the design requirements and the technologies with which to satisfy these requirements, this chapter looks at how the system was implemented. The framework was implemented in two phases; the processing of the user model data phase and the implementation of the coordinated visualisation representation of the user model phase. This chapter first looks at the Client-Server interaction as this takes place in both phases. It then outlines the implementation of both phases individually, providing additional details of the client-server interaction as required. Finally, the inner workings as the user interacts with the system are detailed.

6.1 Client-Server Interaction

Communication is required between the client and the server throughout the generation of and the interaction with user models. This takes place in the form of an AJAX request from the client, which is intercepted and rerouted by the Slim Framework to execute the required server code, thus querying the database.

AJAX requests always contain a HTTP method and a URL identifying the resource of interest. It will also contain data where data is required to execute the query due to the stateless nature of the server. The HTTP methods used in this framework are:

- GET to retrieve data from the server.
- PUT to update data on the server or retrieve data from the server when data is required to perform the query.
- POST is used to create new tables.

The resources of interest are the XML file, the semantic data tables and the user model tables. Further description of the methods will be provided as they arise in the implementation.

6.2 The User Model

6.2.1 The CULTURA Data Store

Log and semantic data were obtained from the CULTURA project to develop the framework. The log data was contained in an XML document with log entries in the form of:

```
<sensorrawdata>
  <date timestamp="2013-09-19 18:13:38.942"/>
  <context sourceid="culturasystem-1641"/>
  <participant id="605"/>
  <predicate tag="pagevisit"/>
  <valuedata pagetype="content" pageid="826112r116" navigationsource=""/>
  <data isinvalid="false"/>
</sensorrawdata>
```

The XML document was used in lieu of a database for the development of the framework.

The semantic data was collected in the form of an SQL data dump. MySQL supports the importing of .sql files, replicating the database from which the data came. The data was imported into a MySQL database and produced 8 tables, linked as illustrated in the relational schema in Figure 6.2.1.1. The highlighted attributes are the attributes that were used to generate the user model, as will be outlined in section 6.2.3.

6.2.2 Extracting the Log Data

Data is extracted from the log data the first time a user uses the user model visualisation system. The data is used to generate the user model data which is stored in the user model database. If used on a live system, user models could be updated at every interaction so it is not unreasonable to assume that the user model data would be stored for future use.

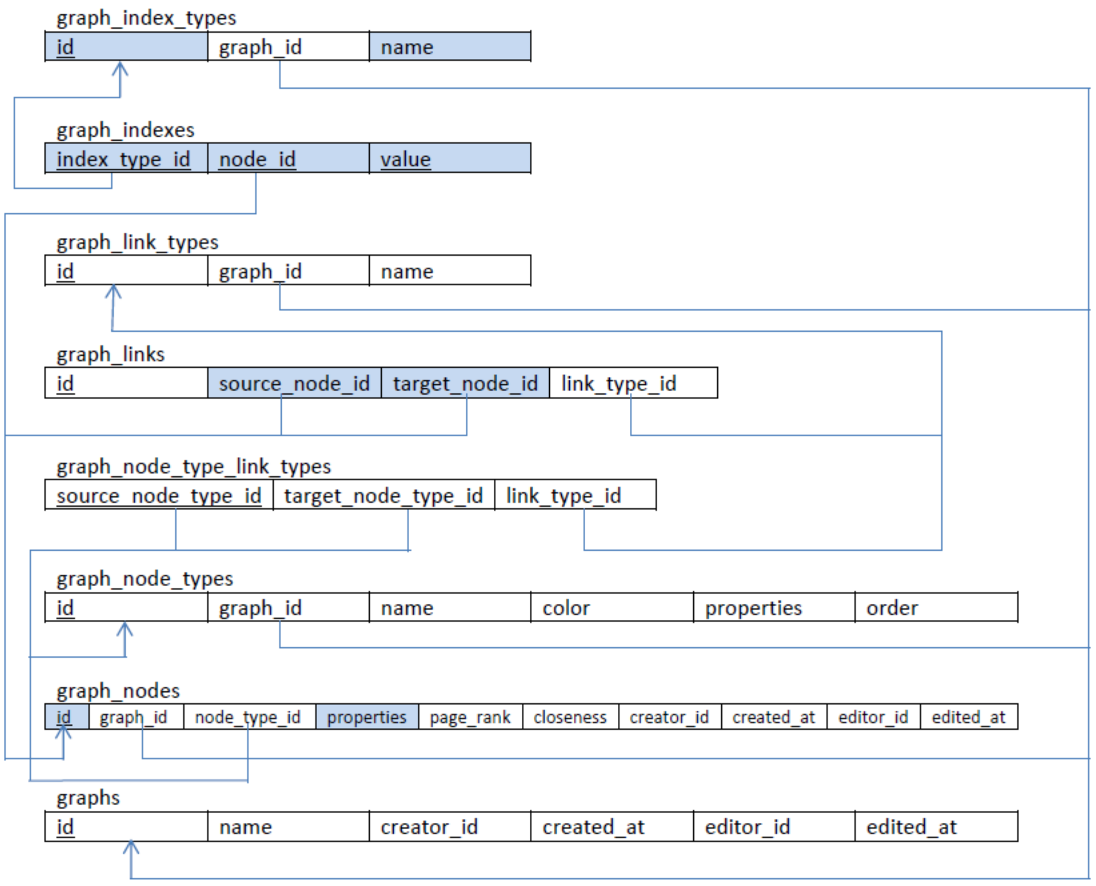
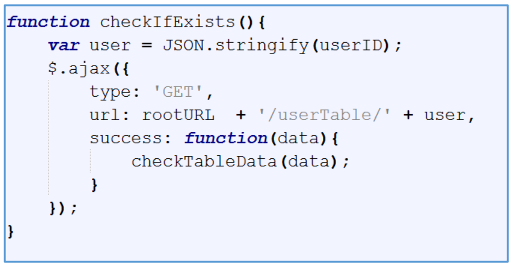


Figure 6.2.1.1: Relational Schema for the CULTURA semantic data

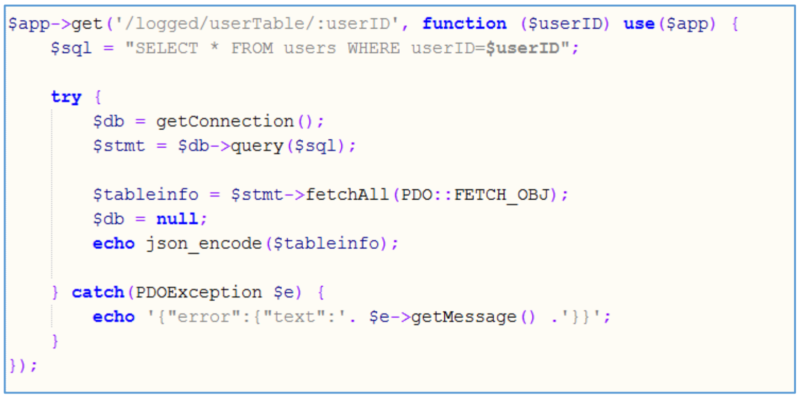
On loading the HTML page, the user is asked to enter an ID. A GET method is used by the client to check whether the user data has been processed, as illustrated in figure 6.2.2.1.



```
function checkIfExists(){
    var user = JSON.stringify(userID);
    $.ajax({
        type: 'GET',
        url: rootURL  + '/userTable/' + user,
        success: function(data){
            checkTableData(data);
        }
    });
}
```

Figure 6.2.2.1: checkIfExists function with AJAX request using GET method.

The GET method only allows the inclusion of data in the request if it is appended to the URL, therefore the user ID is appended to the URL. The Slim Framework reroutes the request to the anonymous function matching the method and URI. As can be seen in the first line of Figure 6.2.2.2, the Slim Application points to a ‘get’ function where the end of the path matches the end of the URL in Figure 6.2.2.1.

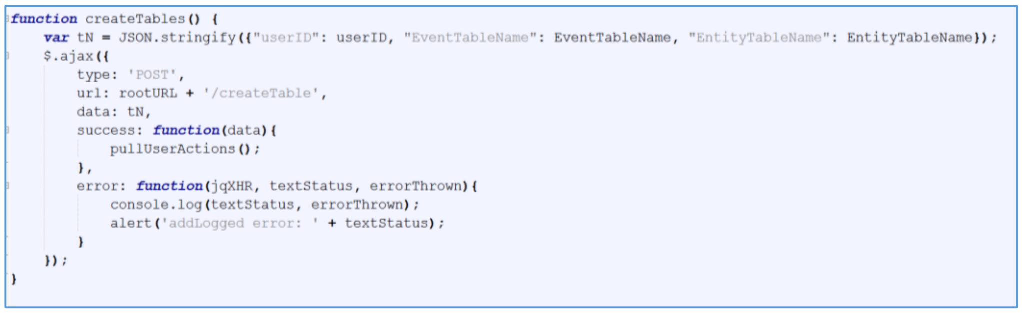


```
$app->get('/logged/userTable/:userID', function ($userID) use($app) {
    $sql = "SELECT * FROM users WHERE userID=$userID";

    try {
        $db = getConnection();
        $stmt = $db->query($sql);

        $tableinfo = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        echo json_encode($tableinfo);

    } catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
});
```

Figure 6.2.2.2: Anonymous function identified through the HTTP GET method and resource.

In the anonymous function, the SQL query is prepared in a statement using the data from the client to set the criteria. A PDO is used to connect to the database and to then query the database. All data satisfying the query is selected and returned as objects into the $tableinfo variable. The database connection is terminated by setting it to null. The data to be returned to the client is encoded in JSON format and echoed back to the callback function. The data is then used by the client to complete further actions. As the userID is the Primary Key for that table, at most one record can exist. If the JSON object is empty, this signifies that the data has not been processed.

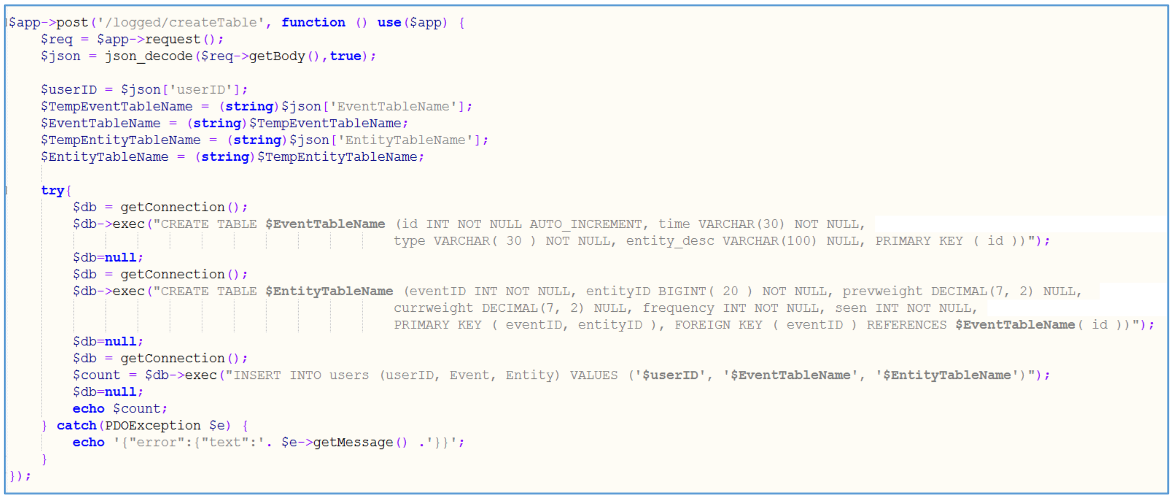
If the user model data has been stored, the client requests the user data and generates the visualisation as will be outlined in section 6.3. Otherwise, the user log data for that user must be processed. The client starts by creating two tables for the user – an events table and a user model entity table. This is achieved through the use of a POST method as illustrated in figure 6.2.2.3.



```
function createTables() {
    var tN = JSON.stringify({"userID": userID, "EventTableName": EventTableName, "EntityTableName": EntityTableName});
    $.ajax({
        type: 'POST',
        url: rootURL + '/createTable',
        data: tN,
        success: function(data){
            pullUserActions();
        },
        error: function(jqXHR, textStatus, errorThrown){
            console.log(textStatus, errorThrown);
            alert('addLogged error: ' + textStatus);
        }
    });
}
```

Figure 6.2.2.3: createTables function with AJAX request using POST method.

The POST method allows the inclusion of data wrapped in a JSON string in the AJAX request. In this case the user ID, event table name and entity table name are wrapped in a JSON string. The Slim Framework reroutes the request to the anonymous function matching the method and URL.

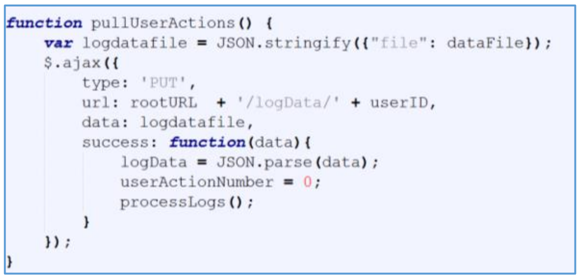


```
$app->post('/logged/createTable', function () use($app) {
    $req = $app->request();
    $json = json_decode($req->getBody(),true);

    $userID = $json['userID'];
    $TempEventTableName = (string)$json['EventTableName'];
    $EventTableName = (string)$TempEventTableName;
    $TempEntityTableName = (string)$json['EntityTableName'];
    $EntityTableName = (string)$TempEntityTableName;

    try{
        $db = getConnection();
        $db->exec("CREATE TABLE $EventTableName (id INT NOT NULL AUTO_INCREMENT, time VARCHAR(30) NOT NULL,
                                                type VARCHAR( 30 ) NOT NULL, entity_desc VARCHAR(100) NULL, PRIMARY KEY ( id ))");
        $db=null;
        $db = getConnection();
        $db->exec("CREATE TABLE $EntityTableName (eventID INT NOT NULL, entityID BIGINT( 20 ) NOT NULL, prevweight DECIMAL(7, 2) NULL,
                                                currweight DECIMAL(7, 2) NULL, frequency INT NOT NULL, seen INT NOT NULL,
                                                PRIMARY KEY ( eventID, entityID ), FOREIGN KEY ( eventID ) REFERENCES $EventTableName( id ))");
        $db=null;
        $db = getConnection();
        $count = $db->exec("INSERT INTO users (userID, Event, Entity) VALUES ('$userID', '$EventTableName', '$EntityTableName')");
        $db=null;
        echo $count;
    } catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
});
```

Figure 6.2.2.4: Anonymous function identified through the HTTP POST method and resource.

The anonymous function decodes the data and uses it to create the three PDOs required to create the two tables and to update the users table by adding an entry containing the user ID, event table name and user model entity table name, as illustrated in Figure 6.2.2.4. The number of rows updated in the users table is returned to the client through the $count variable, indicating a success if 1 is returned.

On notification that the tables have successfully been created and that the user has been added to the table of users, the client requests the log data for that user from the server through the use of a PUT method. The PUT method is used when sending data in the form of a JSON string which the server needs to retrieve related data. The server then returns the data to the client as a JSON string. In this case, two forms of data are send – the user ID is appended to the end of the URL *and* the name of the log data file is sent as a JSON string, as can be seen in Figure 6.2.2.5.



```
function pullUserActions() {
    var logdatafile = JSON.stringify({"file": dataFile});
    $.ajax({
        type: 'PUT',
        url: rootURL  + '/logData/' + userID,
        data: logdatafile,
        success: function(data){
            logData = JSON.parse(data);
            userActionNumber = 0;
            processLogs();
        }
    });
}
```

Figure 6.2.2.5: pullUserActions function with AJAX request using PUT method.

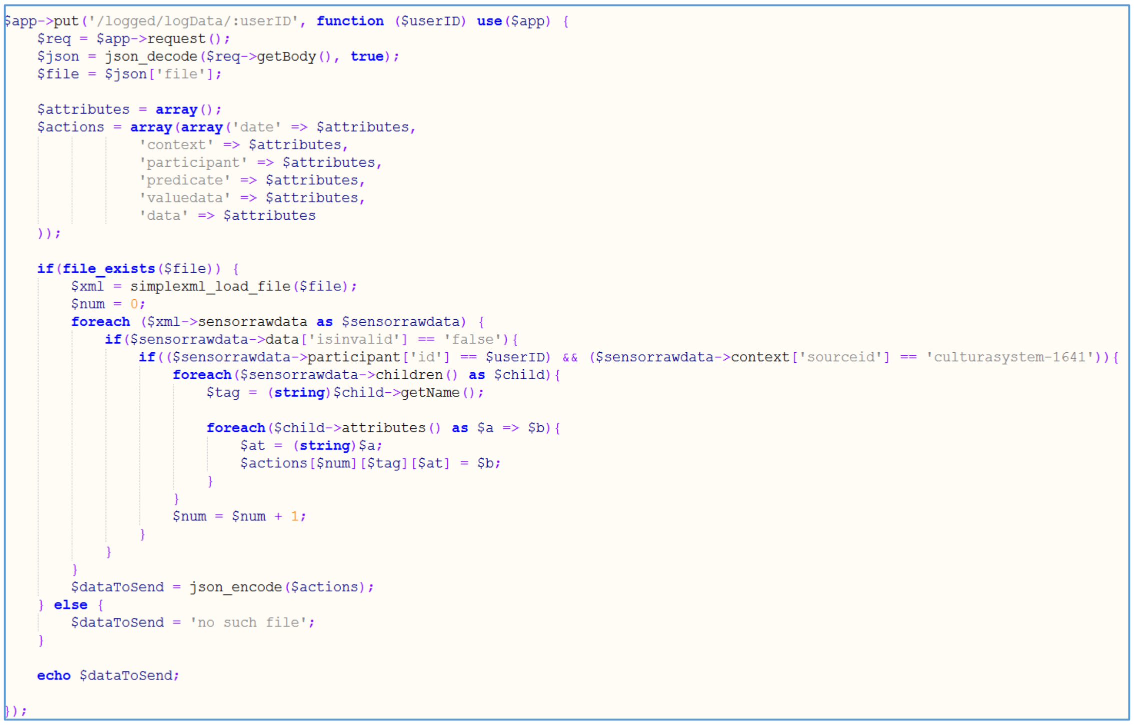
The Slim Framework reroutes the request to the anonymous function matching the method and URI, as seen in Figure 6.2.2.6. The JSON data is decoded to extract the file name. The data in the file is converted into a PHP SimpleXMLElement Object. A PHP script iterates through the elements, checking the value of the elements' children to determine whether the data is valid, whether it belongs to the required CULTURA collection and if the user ID matches that of the current user. In section 6.2.1, a sample log entry was illustrated. Since we are developing the framework using the CULTURA 1641 collection, this record would match the requirements if the user ID was 605 as it contains the children

<context sourceid="culturasystem-1641"/>

and

<data isinvalid="false"/>.

When a relevant element, or user log, has been identified, the full record is added to the results set. Once all elements have been checked, the data is encoded and sent to the client as a JSON string.



```
$app->put('/logged/logData/:userID', function ($userID) use($app) {
    $req = $app->request();
    $json = json_decode($req->getBody(), true);
    $file = $json['file'];

    $attributes = array();
    $actions = array(array('date' => $attributes,
                'context' => $attributes,
                'participant' => $attributes,
                'predicate' => $attributes,
                'valuedata' => $attributes,
                'data' => $attributes
    ));

    if(file_exists($file)) {
        $xml = simplexml_load_file($file);
        $num = 0;
        foreach ($xml->sensorrawdata as $sensorrawdata) {
            if($sensorrawdata->data['isinvalid'] == 'false'){
                if(($sensorrawdata->participant['id'] == $userID) && ($sensorrawdata->context['sourceid'] == 'culturasystem-1641')){
                    foreach($sensorrawdata->children() as $child){
                        $tag = (string)$child->getName();

                        foreach($child->attributes() as $a => $b){
                            $at = (string)$a;
                            $actions[$num][$tag][$at] = $b;
                        }
                    }
                    $num = $num + 1;
                }
            }
        }
        $dataToSend = json_encode($actions);
    } else {
        $dataToSend = 'no such file';
    }

    echo $dataToSend;

});
```

Figure 6.2.2.6: Anonymous function containing PHP script to retrieve log data for a given user ID.

When the client receives the data, it converts it to a JSON object to process the contents. It first checks the length of the object; if there is no data for the particular ID, the user models cannot be generated. If the JSON is not empty, the client loops through the data, identifying user actions from which semantic data can be extracted to feed into the user model.

6.2.3 Extracting the Semantic Data

The user actions from which semantic data can be extracted were identified to be: annotation, bookmark, note, pagevisit and visualisation. Where a user action outside of this set of user actions occurs, the client skips over the interaction. If the user action is within this set, the resource associated with the interaction is used to extract its related entities.

Weights have been assigned to each type of action. The resource associated with the interaction is used to identify related entities. During the implementation, it was identified that the ‘visualisation’ action type can have a number of resources associated with it; this can result in the same entity appearing a number of times for that interaction. Suppose an entity appears 5 times in one visualisation action; this indicates a stronger interest in the entity than in entities that only occur once. Augmenting the weight of the interest by five times the weight associated with the visualisation action type would assign more importance to a visualisation with more than one resource than to a visualisation with only one resource. Augmenting the weight for an interest that appears once by the same amount as an interest that appears 5 times would not be representative of the user’s interests. Instead the weight assigned to each entity in a multiple resource visualisation action is the visualisation weight divided by the number of resources and multiplied by the number of resources the entity is related to in the visualisation action. This means that if five resources are associated with a visualisation and an entity is related to all five resources it will be assigned the full visualisation weight; an entity that is related to only one of the resources will be assigned 0.2 of the weight. The weights for the different action types were chosen to be {"annotation": 7, "bookmark": 5, "note": 4, "pagevisit": 3, "visualisation": 4/nr}, where nr is the number of resources.

Having identified an interaction containing a resource, the client uses the resource identifier to extract the human-readable resource name. This is achieved using a GET method in the same way as the checkExists() function described in section 6.2.1. The data appended to the end of the URL is the resource identifier obtained from the user log. To extract the human-readable resource name, the resource identifier is used to link through the database tables until the required information is retrieved. The query used to achieve this is:

```
SELECT DISTINCT graph_nodes.properties
FROM graph_nodes, graph_indexes
WHERE $entity = graph_indexes.value
AND graph_indexes.node_id = graph_nodes.id
```

As can be seen from the Relational Schema in Figure 6.2.1.1, a tuple is identified in the graph_indexes table by matching the resource identifier ($entity) with the value attribute from that table. The node_id from that tuple is used to identify a tuple in the graph_nodes table by matching graph_indexes.node_id with graph_nodes.id. The properties attribute from the graph_nodes tuple is retrieved and the server returns it to the client.

The client extracts the human-readable name from the data retrieved by the server and logs it in the event table with the time of the interaction and the type of action; this is achieved using a PUT method with a simple INSERT query to update the database.

The client uses the resource identifier to retrieve entities related to the resources; this is achieved using a GET method with the resource identifier appended to the end of the URL. In the anonymous function, the query to extract the required information is:

```
SELECT DISTINCT graph_nodes.id, graph_nodes.properties
FROM graph_nodes, graph_indexes, graph_links
WHERE $depID = graph_indexes.value
AND graph_indexes.node_id = graph_links.source_node_id
AND graph_links.target_node_id = graph_nodes.id
UNION
SELECT DISTINCT graph_nodes.id, graph_nodes.properties
FROM graph_nodes, graph_indexes, graph_links
WHERE $depID = graph_indexes.value
AND graph_indexes.node_id = graph_links.target_node_id
AND graph_links.source_node_id = graph_nodes.id
```

This query first identifies the graph_indexes.node_id for the resource identifier. The graph_links table holds pairs of related entities. The query identifies any pair of entities containing the resource identifier and adds the other entity to the result set. The result set returned to the client is the set of entities related to the resource for the interaction. These are the interests contributing to the user model for the current interaction.

6.2.4 Storing the User Models

Having identified the interests related to a user's interaction, these must be added to the user model, weighted according to the type of action. The user model is initially an empty JSON object which is gradually build up as the client processes each interaction.

To update the user model, the client first checks whether the JSON is empty. If it finds that it is empty, it loops through the related interests, creating a key-value pair where the key is the related interest and the value is the weight for the type of action. The previous weight for that key is set to 0, the frequency is set to 1 and the event at which it was last seen is set to the current event ID.

If it finds that the user model is not empty, it first loops through every interest in the user model setting the previous weight to the current weight value, then loops through again decrementing its weight by 0.5 while enforcing a floor of 0.0 so that weights may not be negative.

It then loops through the related interests checking whether an element exists in the user model which has the related interest as its key. If such an element exists, the value of the weight is incremented by the required weight, the frequency is incremented by 1 and the event at which the entity was last seen is set to the current event ID. If such an element does not exist, the key-value pair is created as before.

As previously mentioned, if the action type is a visualisation, the weights are incremented by the weight divided by the number of resources; interests can be incremented by that value multiple times if they are related to multiple resources. The resources in multi-resource visualisations are handled one at a time; a safe guard has been put in place to ensure the user model is only decayed once while allowing the user model to be incremented repeatedly.

Having fully updated the user model for a particular event, the user model must be stored in the user model database. This is achieved using an AJAX request per user model interest with a PUT method and the user model data for that interest as a JSON string. On receiving the request, the server performs an INSERT on the user model table. Each entry contains the ID of the interest, the previous weight, the current weight, the frequency and the event when the interest was last seen.

Once the user model has been processed for each event, every key in the user model object is an interest that appeared at some stage in the user model. Details relating to the interest are stored separately to avoid duplication. The client uses a GET method with the user model key to retrieve the ID, category and human-readable name. The ID is retrieved for simplification; each element in the JSON object retrieved will contain the ID, so it will not need to be added manually. The category is found in the graph_index_types under the attribute titled “name”. The human readable name is found in the graph_nodes table within the “properties” attribute.

When the client received the entity details from the server, it extracts the human-readable name from the set of “properties”; it then immediately sends another request to the server to store a new entry in the entities table. This entry contains the entity id, the human-readable description and the category to which it belongs.

This concludes the first phase of the implementation; the data for the user model at each interaction over the course of the exploratory search has been computed and can be used to generate visual representations of the user model.

6.3 The Coordinated Visualisation

6.3.1 Requesting the Data

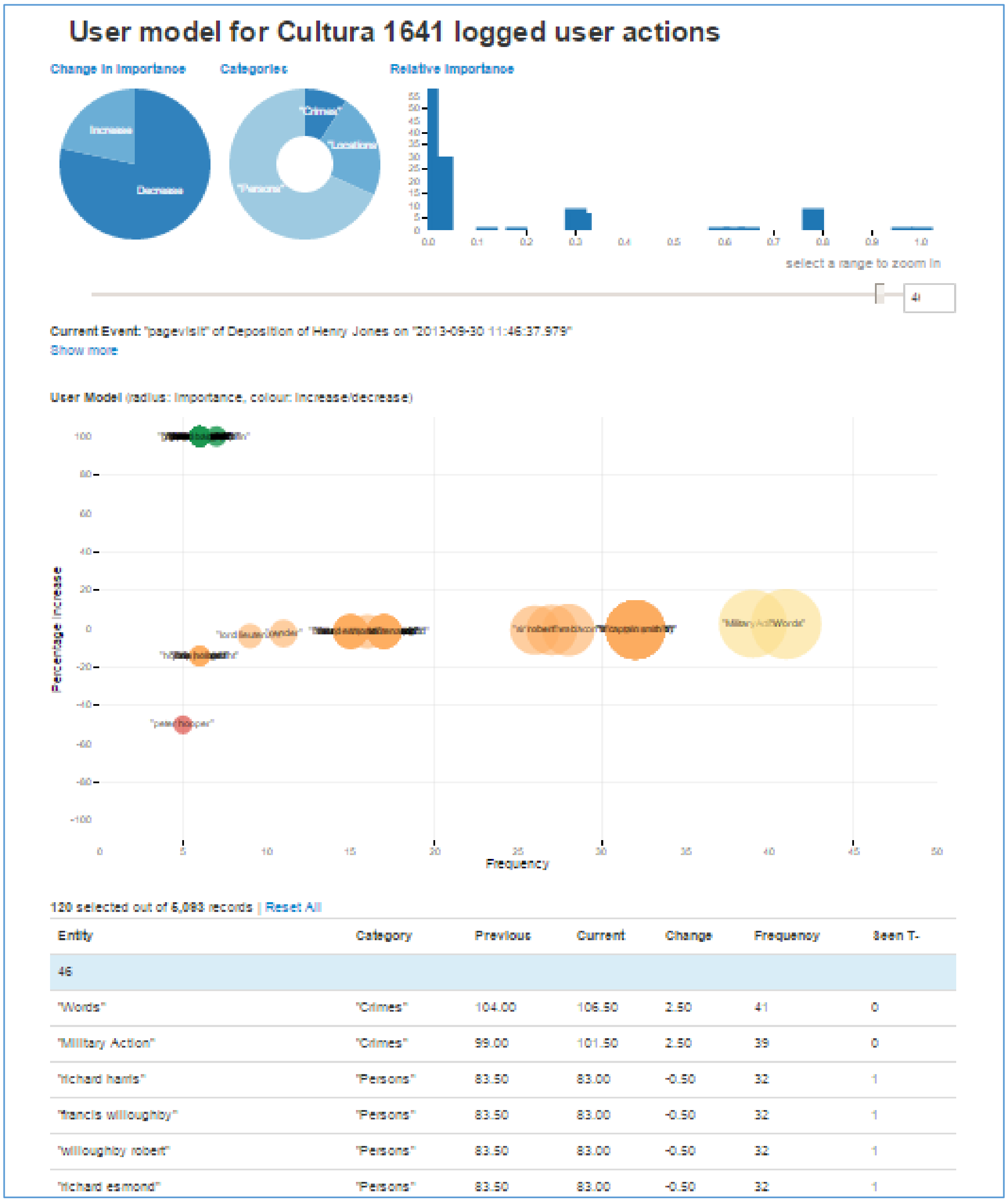
It was initially decided that data would be requested per event. As users changed from one event to another, the data for the new event would be requested from the server; however dc.js does not support this. The Crossfilter must be set up when the JavaScript file is compiled; it can then be filtered on, but it cannot be replaced without reloading the entire page. As a result, when the user logs on and has already been processed, or when a new user logs on and the user model has been fully computed, the full set of data for the user models at each interaction must be retrieved. This is not a problem for the Crossfilter API; as mentioned as in Chapter 4, Crossfilters support high-speed multi-dimensional filtering for datasets containing in excess of 1 million records.

The data is requested from the server using a GET method with the user model table name. The event ID, entity ID, number of times the entity has occurred at that interaction, the previous weight, the current weight and the event ID of the last time the entity was related to the resource are retrieved from the user model table. Using the entity ID from the user model table, the human readable name and the category for that entity are also retrieved. The full set of data is returned to the client and stored in an array of JavaScript objects. This array becomes the Crossfilter. The visualisation dimensions are applied to the Crossfilter and filters, derived data and data bound chart parameter are determined. Filters can be determined within the processing of the visualisation data. For example, categories are not hard-coded in; as a new category is encountered, a new filter is created. This means that the visualisations may be applied to other collections where entities have similar properties.

6.3.2 Visualisations Set-up

The design of the layout for the interface was outlined in the Section 5.3.1. Figure 6.3.2.1 illustrates the final interface. The structure of the page is implemented in the

HTML document through listing the elements in order of appearance and using CSS to set the size, position and font formatting. To render three charts in a row, the charts were assigned to a class and formatting was applied to the class.



User model for Cultura 1641 logged user actions

Current Event: "pagevisit" of Deposition of Henry Jones on "2013-09-30 11:46:37.979"
Show more

User Model (radius: importance, colour: increase/decrease)

120 selected out of 6,093 records | Reset All

Entity	Category	Previous	Current	Change	Frequency	Seen T-
46						
"Words"	"Crimes"	104.00	106.50	2.50	41	0
"Military Action"	"Crimes"	99.00	101.50	2.50	39	0
"richard harris"	"Persons"	83.50	83.00	-0.50	32	1
"francis willoughby"	"Persons"	83.50	83.00	-0.50	32	1
"willoughby robert"	"Persons"	83.50	83.00	-0.50	32	1
"richard esmond"	"Persons"	83.50	83.00	-0.50	32	1

Figure 6.3.2.1: User Interface

The slider and event text box were set up in the HTML document. These appear when the HTML document is loaded; the dc charts however are only rendered once the JavaScript file has been compiled and user model data has been retrieved.

Before the user model can be generated, the event for which the user model should be generated must be selected. The most recent interaction has been selected to be the default user model when the visualisations are first rendered as this is the model users are most likely to wish to view initially. Details of the most recent interaction are therefore retrieved from the server. This is achieved using a GET method containing the table name which uses a query to retrieve the maximum event ID and the corresponding details.

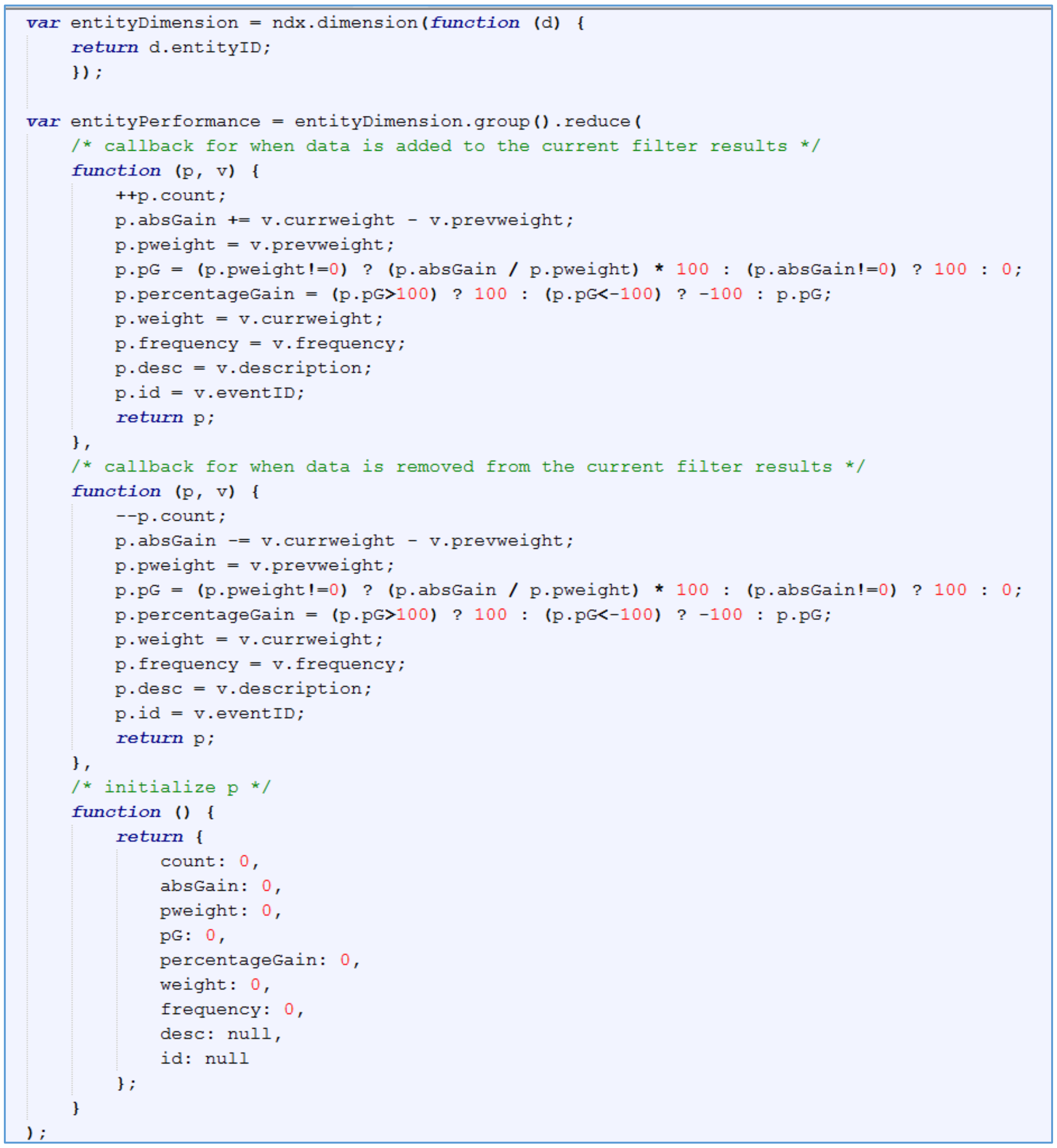
Having retrieved the data set, it must be set up as the Crossfilter data and dimensions must be defined. The dimensions required to produce the interface illustrated in Figure 6.3.2.1 are:

- The event ID
- The entity ID
- The direction of the change in importance
- Category
- Weight

The User Model

A DC Bubble Chart is used to generate the user model. It uses the dimension defined by entity ID, grouped on properties and derived properties of the entity. Figure 6.3.2.2 illustrates the setting up of the dimension and the grouping of the data on required properties of the entity.

‘count’ keeps track of the number of elements in a filtered subset. ‘absGain’ calculates the change in importance. ‘pweight’ holds the value of the previous weight. ‘pG’ calculates the actual percentage change while ‘percentageGain’ caps the value of ‘pG’ at +/-100%. ‘weight’, ‘frequency’, ‘desc’ and ‘id’ hold the values of the current weight, the frequency, the human-readable name and the ID respectively.

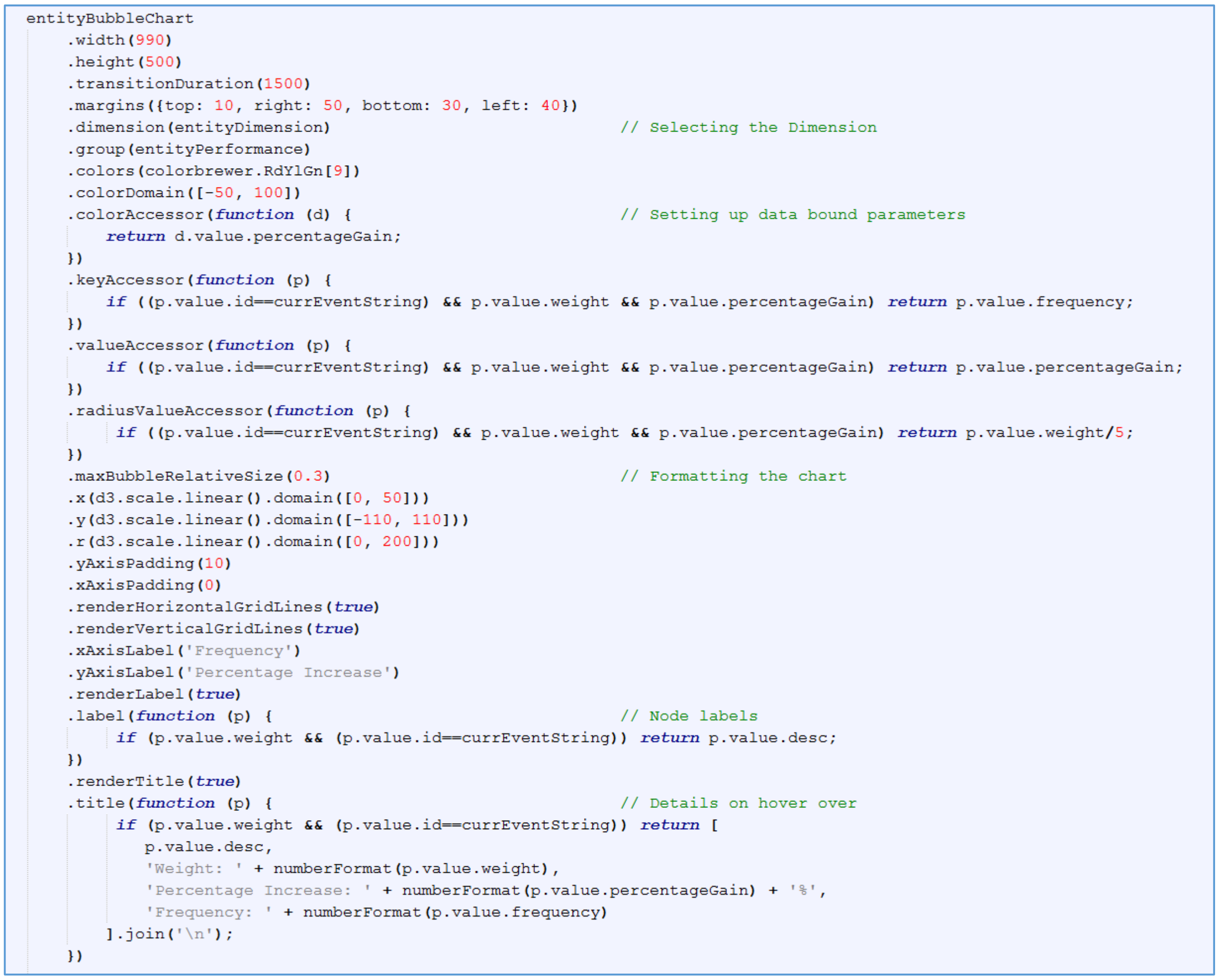


```
var entityDimension = ndx.dimension(function (d) {
    return d.entityID;
    });

var entityPerformance = entityDimension.group().reduce(
    /* callback for when data is added to the current filter results */
    function (p, v) {
        ++p.count;
        p.absGain += v.currweight - v.prevweight;
        p.pweight = v.prevweight;
        p.pG = (p.pweight!=0) ? (p.absGain / p.pweight) * 100 : (p.absGain!=0) ? 100 : 0;
        p.percentageGain = (p.pG>100) ? 100 : (p.pG<-100) ? -100 : p.pG;
        p.weight = v.currweight;
        p.frequency = v.frequency;
        p.desc = v.description;
        p.id = v.eventID;
        return p;
    },
    /* callback for when data is removed from the current filter results */
    function (p, v) {
        --p.count;
        p.absGain -= v.currweight - v.prevweight;
        p.pweight = v.prevweight;
        p.pG = (p.pweight!=0) ? (p.absGain / p.pweight) * 100 : (p.absGain!=0) ? 100 : 0;
        p.percentageGain = (p.pG>100) ? 100 : (p.pG<-100) ? -100 : p.pG;
        p.weight = v.currweight;
        p.frequency = v.frequency;
        p.desc = v.description;
        p.id = v.eventID;
        return p;
    },
    /* initialize p */
    function () {
        return {
            count: 0,
            absGain: 0,
            pweight: 0,
            pG: 0,
            percentageGain: 0,
            weight: 0,
            frequency: 0,
            desc: null,
            id: null
        };
    }
);
```

Figure 6.3.2.2: Setting up the dimension and grouping on properties.

Figure 6.3.2.3 illustrates the code required to set up the bubble chart. Comments have been included to identify where the dimension was set, where the data bound parameters were set up, where the chart was formatted, where the node labels were set up and where the box of additional details which appears when the user hovers over a node is set up.

The percentage change in importance is used to determine the colour of the node; the greener the node, the greater the increase in importance, the redder the node, the greater the decrease in importance. The x-axis value is the frequency of the entity; the y-axis value is the percentage change in importance of the entity; the radius of each node is a fraction of the weight of the entity. These values are only assigned to bubble chart nodes if the entity has a non-zero weight and a non-zero percentage change in value; otherwise the node must not be present in the user model.



```
entityBubbleChart
    .width(990)
    .height(500)
    .transitionDuration(1500)
    .margins({top: 10, right: 50, bottom: 30, left: 40})
    .dimension(entityDimension)                         // Selecting the Dimension
    .group(entityPerformance)
    .colors(colorbrewer.RdYlGn[9])
    .colorDomain([-50, 100])
    .colorAccessor(function (d) {                       // Setting up data bound parameters
        return d.value.percentageGain;
    })
    .keyAccessor(function (p) {
        if ((p.value.id==currEventString) && p.value.weight && p.value.percentageGain) return p.value.frequency;
    })
    .valueAccessor(function (p) {
        if ((p.value.id==currEventString) && p.value.weight && p.value.percentageGain) return p.value.percentageGain;
    })
    .radiusValueAccessor(function (p) {
         if ((p.value.id==currEventString) && p.value.weight && p.value.percentageGain) return p.value.weight/5;
    })
    .maxBubbleRelativeSize(0.3)                         // Formatting the chart
    .x(d3.scale.linear().domain([0, 50]))
    .y(d3.scale.linear().domain([-110, 110]))
    .r(d3.scale.linear().domain([0, 200]))
    .yAxisPadding(10)
    .xAxisPadding(0)
    .renderHorizontalGridLines(true)
    .renderVerticalGridLines(true)
    .xAxisLabel('Frequency')
    .yAxisLabel('Percentage Increase')
    .renderLabel(true)
    .label(function (p) {                               // Node labels
         if (p.value.weight && (p.value.id==currEventString)) return p.value.desc;
    })
    .renderTitle(true)
    .title(function (p) {                               // Details on hover over
         if (p.value.weight && (p.value.id==currEventString)) return [
            p.value.desc,
            'Weight: ' + numberFormat(p.value.weight),
            'Percentage Increase: ' + numberFormat(p.value.percentageGain) + '%',
            'Frequency: ' + numberFormat(p.value.frequency)
        ].join('\n');
    })
```

Figure 6.3.2.3: Setting up the user model visualisation.

The x and y axis were formatted, assigning scales, ranges, padding and labels. The DC API gives the option of varying the ranges of the axes in line with the data displayed; however this was found to hinder the comparison of user models for different events. For example, nodes could appear higher in the chart while declining in importance from one event to the next due to a change in the range of the y-axis. The human-readable description, weight, percentage increase and frequency are displayed on hovering over the node.

Relative Importance Visualisation

A DC Bar Chart is used for the Relative Importance visualisation. This visualisation requires the maximum weight of all entities currently presented and divides the weight of each entity by the maximum weight. To achieve this, two weight dimensions were defined. The first returns the current weight of elements in the dataset while searching for the maximum weight for each event; the second uses the maximum weight for each event determined by the first dimension and returns the ratio of the current weight of an element to the maximum weight of the event it belongs to. The second dimension is then grouped to return the number of elements at each observed ratio. Having defined the dimensions and groupings, the bar chart was set up. The size of the chart and the number of bins required were set up. By default the chart takes the count in each bin to be the height of the individual bars. The ability to select a range is also applied by default in DC Bar Charts.

The Filtering Tools

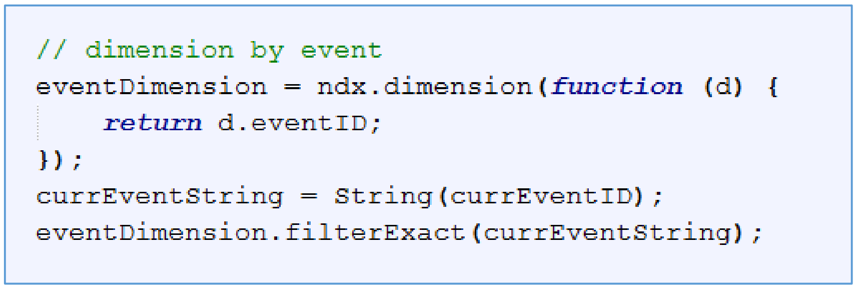
DC Pie Charts are used for the Change in Importance and Categories visualisations. The dimension for the Change in Importance visualisation returns either ‘Increase’ or ‘Decrease’, derived by calculating the change in weight and returning ‘Increase’ for values greater or equal to zero and ‘Decrease’ for values less than zero. The Category visualisation returns the category attribute of the dataset. These are not hard-coded; a slice is defined by the API for each category encountered. In both visualisations, the dimension is grouped to count the number of elements in each slice of the dimension. The size of the slice is determined by the proportion of the total number of elements in each slice. The label on each slice is the value returned when the dimension is defined; that is Increase and Decrease for the Change in Importance visualisation, and the Category names for the Categories visualisation.

The Data Table

DC supports the use of data tables in coordinated visualisations to illustrate the data contained in the filtered dataset; however it does not support interactions with the data table or filtering using the data table. The first weight dimension described in the Relative Importance section above is the dimension used for the data table; this enables the listing of table entries by decreasing weight order. The column headings and properties or derived properties corresponding to the headings were set up.

The Event Dimension

The DC API enables interactions with coordinated visualisations, particularly to support the selection of subsets of data through selecting one or more filters to be applied to visualisations; however it lacks features to enforce the selection of one filter only. To overcome this issue, the event dimension is defined and filtered using the current event as illustrated in figure 6.3.2.4. While subsequent dimensions and groupings are performed over elements for all events, only the elements for the current event will be displayed in the visualisations.



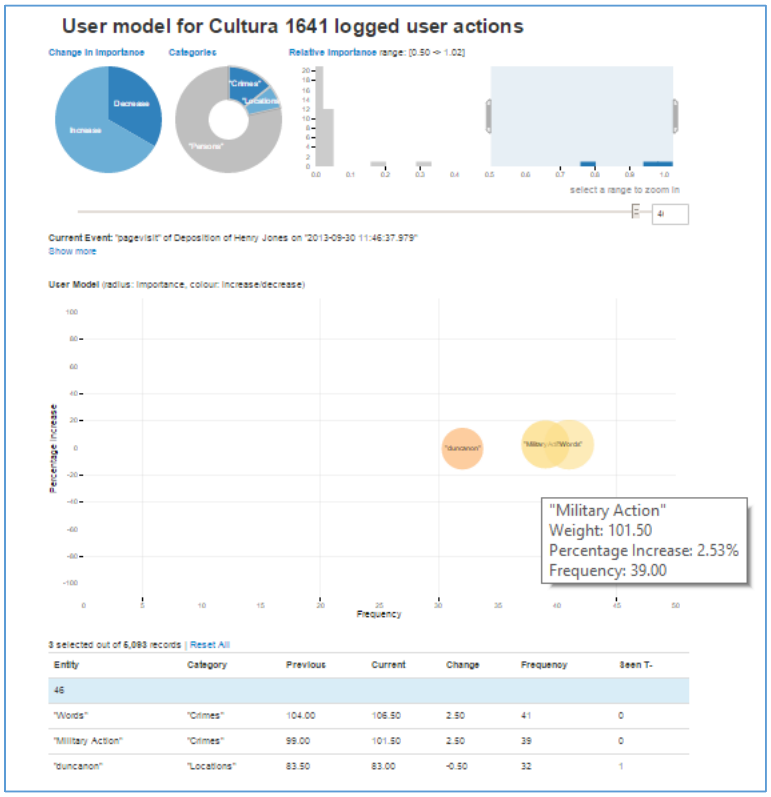
```
// dimension by event
eventDimension = ndx.dimension(function (d) {
    return d.eventID;
});
currEventString = String(currEventID);
eventDimension.filterExact(currEventString);
```

Figure 6.3.2.4: Event dimension filtered by current event ID.

To change the event, a slider and text box were provided as HTML elements. jQuery was used to trigger event handlers when either element was interacted with. The event handlers mirror each other. First, the event handler updates the value of the other element. The filter currently on the event dimension is removed, the new current event is applied as a filter on the event dimension and all charts are redrawn. Some issues arose however when the event was changed with filters applied on the visualisations. These issues and the measures taken to overcome them are described in the next section.

6.4 Generating the User Models

With the visualisations set up as outlined in the previous section, having retrieved the data, the overview of the current user model could be presented to users as illustrated in figure 6.3.2.1. Users could then zoom in on a particular tier of interests, filter the data set to view subsets of the data in terms of categories and/or interests related to or not related to the current resource. Additional details were available from the data table or on hovering over nodes. Figure 6.4.2.1 illustrates the interface looking at interest above a minimum threshold of half the maximum importance for that event, filtered to only view interests from the Crimes and Locations categories, with additional details provided in the data table and on hovering over a node such as the “Military Action” node.


User model for Cultura 1641 logged user actions
Change in importance
Categories
Relative importance range: [0.50 -> 1.02]
Decrease
Increase
"Crimes"
"Locations"
"Persons"
select a range to zoom in
User Model (radius: importance, colour: increase/decrease)
Percentage increase
Frequency
"duncanon"
"Military Action"
"Words"
Show more
Reset All

Current Event: "pagevisit" of Deposition of Henry Jones on "2013-09-30 11:46:37.979"

"Military Action"
Weight: 101.50
Percentage Increase: 2.53%
Frequency: 39.00

3 selected out of 6,093 records | Reset All

Entity	Category	Previous	Current	Change	Frequency	Seen T-
46						
"Words"	"Crimes"	104.00	106.50	2.50	41	0
"Military Action"	"Crimes"	99.00	101.50	2.50	39	0
"duncanon"	"Locations"	83.50	83.00	-0.50	32	1

Figure 6.4.2.1: Zoom and filter, then details-on-demand.

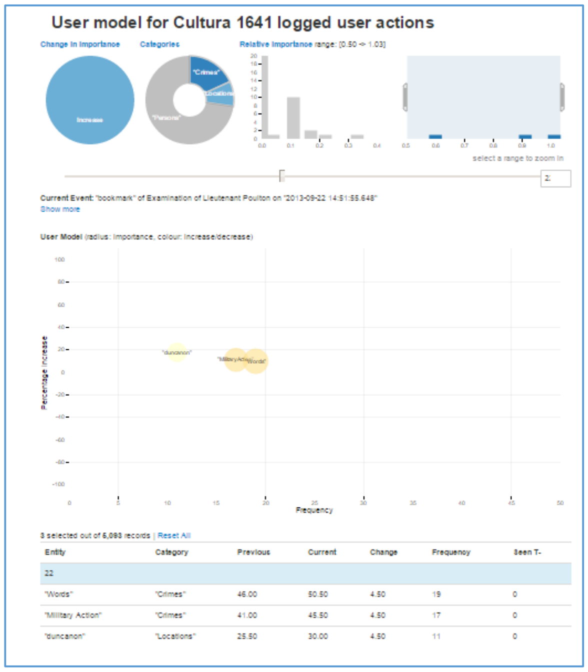
The framework was found to work well if the user only filtered using events, or only using the coordinated visualisations. When moving from one event to another, the use of other filters, even if all filters were removed before the user changed to a different event, could lead to errors in the entities which were presented in the user model. For example, if the event was changed while the 'Increase' filter was applied, the user model was sometimes rendered with the 'Decrease' filter applied instead; that is filters were reversed. This is due to the manner in which filters are applied by the DC API. Filters are not removed but rather they are reapplied. When the charts were redrawn, the filters were reapplied, thus removing the required filters and applying the unwanted filters. This also caused some ambiguity around all filters being removed as opposed to all filters being applied. When the visualisation is loaded no filters are applied. If the 'Increase' filter is applied followed by the 'Decrease' filter, the filter on the chart is [Increase, Decrease]. To the user, it looks the same whether all filters or no filters are applied; however the data is handled differently by the system in each scenario. Changing the data set when no filters appeared to be applied could result in no data being displayed.

To counteract this jQuery was used to log the filters applied to charts when they were interacted with. This involved a slight modification of the dc.js library to facilitate the extraction of the filters applied. The filters were logged in a JSON object using the chart names as keys. When the event was changed, having adjusted the filter on the event dimension, the event handler checked whether filters were applied on each chart. If filters were currently applied, a deep copy of the filters was saved to ensure changes to the charts would not overwrite the copy. The filters were reapplied to the chart, therefore removing the filters. If no filter was currently applied on the chart, all possible filters were applied to the chart. The charts were then redrawn, forcing the recalculation of parameters. The event handler then checked whether deep copies of the filters had been saved for each chart. Where filters had been saved, these were reapplied to the charts. Where no filter had been saved, all possible filters were reapplied. This meant that all charts had the same filters applied to them as when the event was changed, and that all charts had been forced to recalculate all of their parameters for the new subset of data.

Finally, the event handler calls on a function that uses the event table name and new event ID as the data for a PUT method to retrieve the human readable name of the resource the user interacted with, the type of action performed and the date on which the action was performed. These details are displayed above the user model.

6.5 The User Experience

Having implemented the visualisations and the event handler, it was possible to move between events with filters applied as required. Figure 6.5.1 illustrates the user model when a user viewing the zoomed and filtered user model from figure 6.4.2.1 changes the event from event 46 to event 22. Moving through the different events, the user would notice a strong relationship between the location "duncannon" and the crimes "Military Action" and "Words". This indicates that by reflecting over the history of the user model, users may be able to relate interests in subsets of their user model.



Current Event: "bookmark" of Examination of Lieutenant Poulton on "2013-09-22 14:51:55.648"

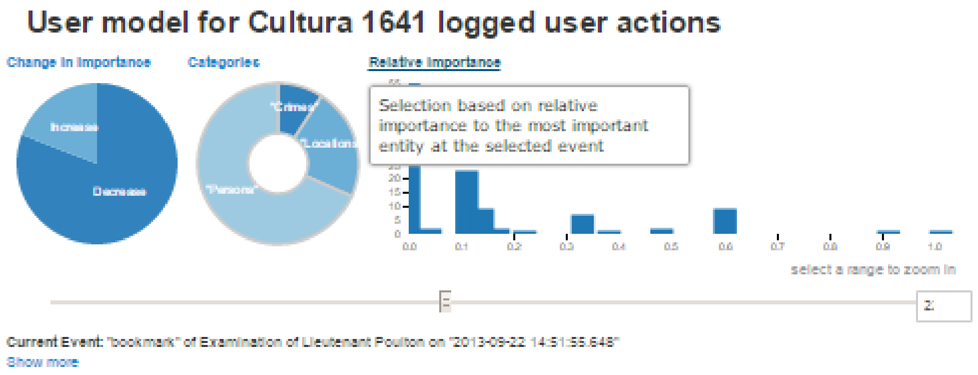
User Model (radius: importance, colour: increase/decrease)

3 selected out of 5,093 records | Reset All

Entity	Category	Previous	Current	Change	Frequency	Seen T-
22						
"Words"	"Crimes"	46.00	50.50	4.50	19	0
"Military Action"	"Crimes"	41.00	45.50	4.50	17	0
"duncanon"	"Locations"	25.50	30.00	4.50	11	0

Figure 6.5.1: Changing events with filters applied.

The chart titles and classifications for the Change in Importance chart used initially during the implementation were revised following feedback regarding their ambiguity to users unfamiliar with the inner workings of the framework. To enhance user experience, the interface should be as self-explanatory as possible. To ensure this was the case, additional information labels were added to the charts, displayed when the user hovers over the chart title, providing a brief description of the purpose of the chart. The labels can provide support to new users and act as a reminder for returning users but will not interfere will seasoned users as they only appear on demand. Figure 6.5.2 illustrates the information label for the Relative Importance chart. This was implemented in the HTML document using jQuery-ui to format the text boxes.


User model for Cultura 1641 logged user actions

Selection based on relative importance to the most important entity at the selected event

Current Event: "bookmark" of Examination of Lieutenant Poulton on "2013-09-22 14:51:55.648"

Figure 6.5.2: Information label for the Relative Importance chart.

The framework had now been designed and implemented in line with the objectives and six of Shneiderman’s steps. The final objective was to design an appropriate task based evaluation and test that the technology works as expected and meets the required goals. The evaluation method and observations are discussed in the next chapter.

7 Evaluation

An appropriate evaluation was carried out to test that the technology works as expected and meets the required goals set at the beginning of this project. This chapter outlines the evaluation method employed to achieve this. The observations from the evaluation are then discussed.

7.1 Evaluation method

In order to demonstrate that the goals had been met, the evaluation needed to demonstrate to following:

- The user interface was user friendly
- The tools provided were useful to view informative subsets of the data
- The user could perform complex analysis of the model
- The system provided a good representation of the user’s interests
- The user was able to reflect on how the user model changed over time
- The user was able to deduce a change in the focus of the exploration over time
- The user was able to identify and differentiate between primary and secondary interests
- The user was able to understand why the system perceived this to be their user model.

This was achieved through a task-based evaluation, a post-task questionnaire and a System Usability Scale (SUS) Questionnaire. These documents be seen in Appendices A, B and C respectively.

The aim of the first few questions of the task-based evaluation was to gently familiarise the participant with the framework while demonstrating the usability of the system in terms of user-friendliness and of technological capability. Participants were asked to retrieve various pieces of information requiring gradually more complex interactions and analysis. As the analysis increased in complexity, the scrutability of the user model and the robustness of the system were tested. Users were required to retrieve information about various subsets of information using the overview first, then using the secondary interests and finally using the secondary interests in an earlier event. This facilitated the testing of the framework's adherence to six of Shneiderman's guidelines – overview first, zoom and filter, details-on-demand, relate and history. These questions make up parts 1, 2 and 3 of the participant tasks which can be viewed in Appendix A.

The tasks in parts 1, 2 and 3 were fact finding tasks, to which there were correct and incorrect answers. The next questions were aimed at testing the reflective capabilities of the user models, in particular through examining the temporal evolution of the user interests. These tasks included comparing user models at specific events and detailing observations made when scrutinising the changes over time.

Some weaknesses were of the technology were identified during the implementation. Tasks were set that would highlight these weaknesses, in particular to see how users would cope with the weaknesses and how the weaknesses would interfere with the completion of the tasks.

Finally, participants were set a task to evaluate their ability to determine, through reflection over the user model, how the user model was being generated by the system.

The tasks set were designed to assess the ability of the system to present, and allow the user to draw conclusions on, subsets of interest which would typically be viewed during reflection with the aim of extracting a deeper understanding of the data. For example users were asked to identify persons and locations in the middle tier of the data in the current event, then compare the results to the same subset of data in an earlier event.

Having completed the task-based evaluation, participants completed a post-task questionnaire in the style of an interview. The valence of the questions was alternated at every question to prevent systematic answers. The purpose of questionnaire was to gauge how user-friendly and useful participants found the various data representation in the coordinated visualisation to be and how self-explanatory and easy to understand they found the titles and labels used in the interface. By using an interview style, participants provided more information about what aspects they particularly liked and disliked than they would have had they simply ticked the boxes. This enabled the identification of additional features that users would find beneficial.

Lastly the SUS questionnaire was completed by the participants. The purpose of the SUS was to get a “quick and easy” indication of the usability of the framework.

7.2 Procedure

The evaluation was carried out in May 2005. Five volunteers undertaking a qualification in a computer-related disciplines partook in the evaluation. Each session lasted approximately 40 minutes. During this time, participants completed a task-based evaluation, a post-task questionnaire and a SUS questionnaire.

Throughout the task-based evaluation, participant behaviour was observed to determine whether participants were seen to struggle with the use of the technology and required assistance in completing a task; however no intervention was required at any time. The post-task questionnaire was completed in the style of an interview. Finally participants completed the SUS questionnaire.

The data from the individual evaluations was recorded and aggregated and the aggregation was analysed. The observations are discussed in the next section.

7.3 Results and Observations

The results from the task-based evaluation and the post-task questionnaire were analysed. All participants were able to perform the fact-finding tasks and extract the required information using the tools provided. Participants were observed to adapt to using the tools very quickly and generally reported finding the tools facilitated the retrieval of the information required.

In part 1, users were asked to extract information from the full set of data for the current event. This involved the use of the Categories and the Change in Importance to filter the data. In the post-task questionnaire, users were asked about the user-friendliness of the filtering tools and the value added from using the tools. The general consensus from the post questionnaire was that the filtering tools were useful for the analysis of subsets of the data and easy to use in terms of selecting pie slices required. When asked about the titles and labels, these were found to be self-explanatory; however some overlap between the labels made them difficult to read at times.

In the previous chapter, the problems arising from the way the DC API removes filters by reapplying filters were discussed. It was mentioned that if a user applied an Increase filter followed by a Decrease filter the user would perceive it as no filter being applied while the system would hold both Increase and Decrease in the filter set. If the user clicked on the Increase slice again to re-select it as before, this time the slice would be de-selected and only Decrease data would be presented. One user specifically commented on the fact that he found it somewhat confusing at times when trying to select slices as he didn’t know whether clicking on a pie slice would select or de-select it.

In part 2, users were asked to look at secondary interests and identify any locations present in the user model. This required the use of the zoom tool, as well as the filtering tools from part 1. Some participants immediately identified from the filtering tool that no locations were present; however some participants were somewhat confused by that fact that while the ‘Location’ slice had contracted to just a line, the line was still visible. This left them unsure as to whether there actually was a location present on examining the filtering tool. In all cases participants did identify that there were no locations, although some users needed to consult the data table to dissipate their uncertainty. While this did indicate a source of ambiguity in the visualisation, it also demonstrated that users had adapted to using the system in a short period of time, by the fact that they immediately scrutinised the categories filtering tool to assess whether a particular category was contained in the model.

The feedback during the post questionnaire was that the zoom were useful for the analysis of subsets of the data, easy to use in terms of selecting the range and had self-explanatory titles and labels.

Part 3 asked participants to perform analysis on an earlier task; this required the use of the slider and/or text box to select an event. Participants were observed to have no problems moving from one event to another; this was reflected in the feedback from the post-task questionnaire. The slider and text methods of selecting events were found to be user-friendly and participants found it easy to navigate to a specific event; however this required knowing the event number in advance. One user suggested that displaying an information bubble while hovering over the slider to identify the details of the event at each position along the slider would be greatly beneficial.

Parts 1-3 illustrated the system developed generated scrutable user models in line with the coordinated visualisations guidelines proposed by Shneiderman to support the exploration of large data sets.

Part 4 asked users for their observations on comparing two user models. It was expected that users would identify a divergence between two of the locations in the user model. In the earlier model, both locations were similar in importance and appeared in the middle tier. In the later model one interest had significantly increased in importance and moved from the middle tier to the top tier, while the other interest had moved from the middle tier to the bottom tier. Most users identified that the interest that was increasing in importance had moved to the top tier. Some users detected the move to the bottom tier by the interest that decreased in importance. Most users also observed that a person examined in an earlier fact finding task had increased importance but remained in the middle tier.

When asked in part 5 to navigate through events from an early event to the most recent event, most participants noticed that as they cycled through the events a group of interests pulled ahead and became more defined importance-wise. One participant specifically made reference to a split between the group of key interests and the group of secondary interests seen in events occurring after the 40th event.

Parts 4 and 5 illustrate the reflective capabilities of scrutable user models portrayed through the use of coordinated visualisations.

The DC API does not support the facility to reduce the data illustrated using a text search. Questions 6 was designed to highlighted this weakness by asking participants to find all occurrences of “Robert”; however participants immediately used CTRL+F to find instances of the required text in the table. Participants were then asked in question 8 to select the “Robert” with the lowest weight in the user model. They observed that the data could be filtered on the user model by clicking on nodes; however they found certain nodes could not be selected due to belonging to a cluster of nodes obstructing the node they required. This conclusion was as expected; the question had been specifically formulated to highlight these issues with the user model.

Questions 6 and 7 required the use of the user model visualisation and the data tables. The post-task questionnaire determined that the user model visualisation was the only visualisation which scored below average on all features assessed in the post questionnaire. The user model visualisation scored slightly below average in terms of user-friendliness. The overlapping between the nodes was found to hinder the ability to select individual interests. Participants were able to identify that there were nodes present in the user model with a certain relative importance; however they found that it was difficult to identify from the model which nodes were present if they were obstructed by other nodes. The filtering tools were found to assist with this problem in some circumstances; however where nodes from the same category overlapped participants needed to rely on the tables. Users reported liking the ‘on hover details-on-demand’, but again found it hard to hover over some nodes due to overlapping nodes.

The post-task questionnaire also determined that the data table was found to provide much insight into the data in the user model when nodes overlapped; however participants would have liked further clarification of the meaning of the headings. It was suggested that supporting search and filtering on the list would be beneficial, as would the ability to freeze headings so that they would always be visible. Unfortunately these features are not supported by the DC API.

Most users were able to identify that “annotation” generated the largest increase in importance; however some users were unsure how the weight was calculated and assigned to the interests in the user model.

The framework received a SUS score of 67. Anything above 68 is considered above average; it can be inferred that the usability of the system was about average. Due to the low number of participants, however, no further information can be drawn from the SUS score.

The fact that participants were able to circumnavigate the temporal span of the user model, performing complex analyses and retrieving the required information demonstrates the robustness of the framework and the support it can provide to yield deeper insight into the user model data.

As a whole, the framework was found to be technologically sound and user friendly. The framework met the scrutability criteria and adhered to Shneiderman's visualisation guidelines. Participants were able to extract the require information and adapted to relying on the tools very quickly. Through the use of the tools, participants were able to view informative subsets and perform complex analyses on the model to determine the user's interest in any user model, reflect of over the evolution of the model over time and deduce changes in the focus of the exploration; however a different choice of visualisation for the user model may have been more insightful. Most participants were able to identify and differentiate between primary and secondary clusters of interests. Users were able to identify the type of action that resulted in the largest increase in importance; however they were unsure how the weights were accumulated.

From these observations, it can be seen that the system generally satisfies the requirements listed at the beginning of this chapter. In doing so, the goals set out for this project were achieved. These were:

- Store sufficient information about the user model at each event to be able to recreate all historical user models as well as the current user model.
- Present a user interface that is easily adopted by users to support reflection.
- Render the user model for any event highlighting changes compared to the previous event and providing useful human-readable details on demand of the changes.
- Facilitate the performance of complex analyses on the user model.
- Provide tools to view useful subsets of the model to gain a deeper insight into the user model.
- Allow the user to throttle the level of information presented to focus on primary interests only.
- Allow the user to throttle the level of information presented to analyse secondary interests so as to identify changes in the focus of their exploration and possible serendipitous relationships.

- Enable users to reflect on how the model and hence their interests have changed over time.

It is worth noting that, while the perceived interests were clear from the model, this does not guarantee that these were the actual interests of the test user. The system, however, is unlikely to perfectly model the user interests; for this reason the user and system should have the facility to collaborate. This project looked at generating a scrutable and controllable user model to support reflection. For the user to collaborate with the system, they must themselves have a good understanding of what these interests are. This evaluation has shown that reflection using a scrutable and controllable user model can enable users to achieve this understanding.

The final chapter concludes that the framework was developed as per the goals set out. It outlines where the framework was found to work particularly well, and where the framework was found to be limited. It will close by looking at some additional features which could be included in the framework, including collaboration between the user and the system.

8 Conclusion

This project proposed to develop a framework to generate scrutable and controllable user models to support reflection using coordinated visualisations. Through reflection users would gain a deeper understanding of the information portrayed in the user model visualisation and of how the information had been generated.

The system developed proved highly successful in imparting this deeper understanding onto users. Complex analysis and reflection over the user model was easily adopted by novice users who succeeded in identifying how the model, and hence the interests, had changed over time. Reflection over interests drifting into and out of the user model allowed users to gain an understanding of the modelling approach utilised by the system, specifically how the system's perception of their beliefs had been derived. Through the throttling of information, users were able to restrict their analysis to primary or secondary interests only, observing changes in the focus of their exploration and facilitating opening themselves up to the discovery of serendipitous relationships.

This research focused on supporting reflection in line with the "reflect" phase of CULTURA's four-phase personalisation approach. In this phase, through reflection over the interests portrayed in the user model, users may identify any misalignment between their believed interests and the system's interpretation of their interests, adjusting the model, as required, to better reflect their interests. Any changes made will be incorporated by the system when making recommendations in the suggest phase. This requires that users understand the cause for the misalignment, which may not be evident through a snapshot overview of the user model.

The user models generated by the framework meet all of the requirements of scrutable user models. In particular, the full implementation of Shneiderman's "overview first, then zoom and filter, details-on-demand, relate and history" guidelines for coordinated visualisation facilitates the presentation of the full set of data held by the system, while allowing the user to throttle the data to make it more manageable, focusing on a particular level of importance. It also facilitated the deeper scutinisation of the throttled data through the application of filters and provision of additional information on demand. Moreover, through enabling reflection over the evolution of the user models, the rationale behind the system's interpretation of a user's interests is made discoverable, as may be serendipitous relationships between interests present in the user model.

This project lays the groundwork for supporting collaboration between the user and the system. An ideal recommender system when exploring voluminous collections would be one that knows exactly what the user wants and can successfully retrieve it; but left to its own devices, technology cannot determine beliefs, it can only process the facts presented to it. Users must fine-tune the user model held by the system to reflect their real needs; but if a user cannot correctly identify their information needs, they cannot direct the system. Users need scrutable and controllable user models that support reflection, and coordinated visualisations can facilitate the scrutability and controllability of the models, as described in this project. With this deeper understanding of their information needs, users will be able to collaborate with the technology efficaciously.

8.1 Future Works

The CULTURA personalisation approach is predicated in part on the reflection of users over their user model, enabling collaboration with the system so as to enhance the support provided by the system during the exploration of the collection. User actions are logged and become part of the modelled process. A snapshot overview of the user model is presented in the form of a tag cloud. The reflective capacity of the user model is limited due to the fact that users cannot see a temporal span of their interests becoming more prominent and drifting away; this prevents reflection over how their interests have changed over time and restricts the understanding of how the user model was generated. The framework was developed using log and semantic data from the 1641 CULTURA collection and has demonstrated its capability in terms of supporting reflection over the user model using this data. It can therefore be integrated into the CULTURA system to work across the 1916 and 1641 originated documents, updating the modelling approach currently in use to provide fuller support to users during their reflection phase.

The use of this framework is not limited to digitalised cultural heritage collections; the information that is most empowering to people is their personal analytics, and emerging technologies have facilitated the collection of the required data. For example, mobile phones can collect information about their owner's location, step count and much more. Personal visual analytics, coupled with reflection and the ability to control the information presented has applications in a number of domains, one of which might be personal health analytics. A computer system monitoring how active or sedentary an individual's lifestyle is could benefit from this framework through enabling reflection over the interpreted view of the system.

Another area which could benefit from the use of this framework is across enterprise communication platforms in the analysis of productivity within different work environments. Through the use of online project coordination tools, such as Slack[2] and Basecamp[3], managers may set assessment metrics on the functioning of their projects. For example, the responsiveness of the team to internal and external queries might be compared to a maximum desired threshold to indicate the performance of their project. In such an application, the framework could be used to provide a visual interface, based on the log data, which enables the exploration and reflection over why the system models a certain variable as it does, and how the model would change if a parameter was adjusted. Such a change in parameter would equivalent to changing the rate at which entities are decayed or the weight assigned to a user action. This would be a means of adjusting the way in which the modelling is performed by the system; however the reflective capability using user models

[2] https://slack.com/
[3] https://basecamp.com/

generated by this framework would benefit users through understanding how adjusting the parameters would affect the overall model representing the functioning of the project.

Bibliography

[AJAX] Asynchronous JavaScript and XML (AJAX). Available online at https://api.jquery.com/jQuery.ajax/.

[Athukorala et al., 14] Athukorala, K., Oulasvirta, A., Glowacka, D., Vreeken, J., Jacucci, G. (2014). *Supporting exploratory search through user modeling.* In UMAP '14: Extended Proceedings: Posters, Demos, Late-breaking Results and Workshop Proceedings of the 22nd Conf. on User Modeling, Adaptation, and Personalization, vol. 1181. CEUR-WS.

[Crossfilter] Crossfilter JavasScript Library. Available online at https://github.com/square/crossfilter/wiki/API-Reference.

[CSS] Cascading Style Sheets (CSS). Available online at http://www.w3.org/Style/CSS/.

[CULTURA] Cultura Outcomes. Available online at http://www.cultura-strep.eu/outcomes.

[D3] Data Driven Documents JavaScript Library (D3). Available online at http://www.d3js.org.

[DC] Dimensional charting JavaScript Library. Available online at http://nickqizhu.github.io/dc.js/.

[Diehl, 07] Diehl, S. (2007). *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. 12th Edition, Chapter 2. Springer, Germany.

[ECMA, 11] ECMA International (2011). *Standard ECMA-262: ECMAScript Language Specification.* ECMA International, Geneva, Switzerland.

[Fielding, 00] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Available online at http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[Gauch et al., 07] Gauch, S., Speretta, M., Chandramouli, A., & Micarelli, A. (2007). *User profiles for personalized information access.* In The adaptive web (pp. 54-89). Springer Berlin Heidelberg.

[Hampson et al., 14] Hampson, C., Lawless, S., Bailey, E., Steiner, C. M., Hillemann, E. C., & Conlan, O. (2014). *Metadata–enhanced exploration of digital cultural collections.* International Journal of Metadata, Semantics and Ontologies, 9(2), 155-167.

[Holub et al., 14] Holub, M., Moro, R., Sevcech, J., Liptak, M. and Bielikova, M. (2014). *Annota: Towards Enriching Scientific Publications with Semantics and User Annotations.* D-Lib Magazine, Vol. 20, No. 11/12.

[Ferrari, 01] Ferrari, M. (Ed.). (2001). *The pursuit of excellence through education*. Routledge.

[Kay, 94] Kay, J. (1994). *The um toolkit for cooperative user modelling.* User Modeling and User-Adapted Interaction, 4(3), 149-196.

[Kay, 99] Kay, J. (1999). *A scrutable user modelling shell for user-adapted interaction*. Basser Department of Computer Science. Sydney, Australia, University of Sydney

[Kay, 06] Kay, J. (2006). *Scrutable adaptation: because we can and must.* Adaptive hypermedia and adaptive web-based systems (pp. 11-19). Springer Berlin Heidelberg.

[Kobsa, 93] Kobsa, A. (1993). *User modeling: Recent Work, Prospects and Hazards.* In Adaptive User Interfaces: Principles and Practice, M. Schneider-Hufschmidt, T., Kühme, and U. Malinowski, (eds.). 1993, North-Holland: Amsterdam.

[Kobsa, 94] Kobsa, A. (1994) *KN-AHS: An Adaptive Hypertext Klient of the User Modelling System.* BGP-MS, Proceedings of 4th Conference on User Modeling.
[LocalStorage] localStorage. Available online http://www.w3.org/TR/webstorage/.

[Koenemann and Belkin, 96] Koenemann, J., and Belkin, N. J. (1996). *A case for interaction: a study of interactive information retrieval behavior and effectiveness.* In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 205-212). ACM.

[Li, 11] Li, W., Ganguly, D. and Jones, G.J.F. (2011). *Enhanced Information Retrieval Using Domain-Specific Recommender Models.* Proceedings of ICTIR 2011, LNCS, vol. 6931, 2011, pp. 201–212.

[Lum, 07] Lum, A. W. K. (2007). *Light-Weight ontologies for scrutable user modelling* (Doctoral dissertation, University of Sydney).

[Ruotsalo et al., 13] Ruotsalo, T., Athukorala, K., Głowacka, D., Konyushkova, K., Oulasvirta, A., Kaipiainen, S., Kaski, S. and Jacucci, G. (2013). *Supporting exploratory search tasks with interactive user modeling.* Proceedings of the American Society for Information Science and Technology, 50(1), 1-10.

[HTML] HyperText Markup Language (HTML). Available online at http://www.w3.org/standards/webdesign/htmlcss.

[HTTP] Hypertext Transfer Protocol Request (HTTP Request). Available online at http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html.

[jQuery] jQuery JavaScript Library. Available online at http://www.jquery.com.

[JSON] JavaScript Object Notation (JSON). Available online at https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

[Marchionini, 06] Marchionini, G. (2006). *Exploratory search: from finding to understanding.* Communications of the ACM, 49(4), 41-46.

[Oracle, 13] Oracle (2013). *Top 10 Reasons to choose MySQL for Next Generation Web Applications.* A MySQL Whitepaper, October 2013. Available online at http://mysql.com/why-mysql/white-papers/.

[PHP] PHP: Hypertext Preprocessor (PHP). Available online at http://www.php.net.

[Shneiderman, 96] Shneiderman, B. (1996). *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations.* University of Maryland, USA.

[Singer et al., 11] Singer, G., Norbisrath, U., Vainikko, E., Kikkas, H. and Lewandowsk, D. (2011). *Search-Logger – Analyzing exploratory search tasks.* Proceedings of the 2011 ACM symposium on applied computing, SAC ’11. New York: ACM, 2011, pp. 751–756.

[Sweetnam, 13]. Sweetnam, M., O Siochru, M., Agosti, M., Manfioletti, M., Orio, N. and Ponchia, C. (2013). *Stereotype or Spectrum: Designing for a User Continuum.* Proceedings of ENRICH 2013 - SIGIR 2013 Workshop.

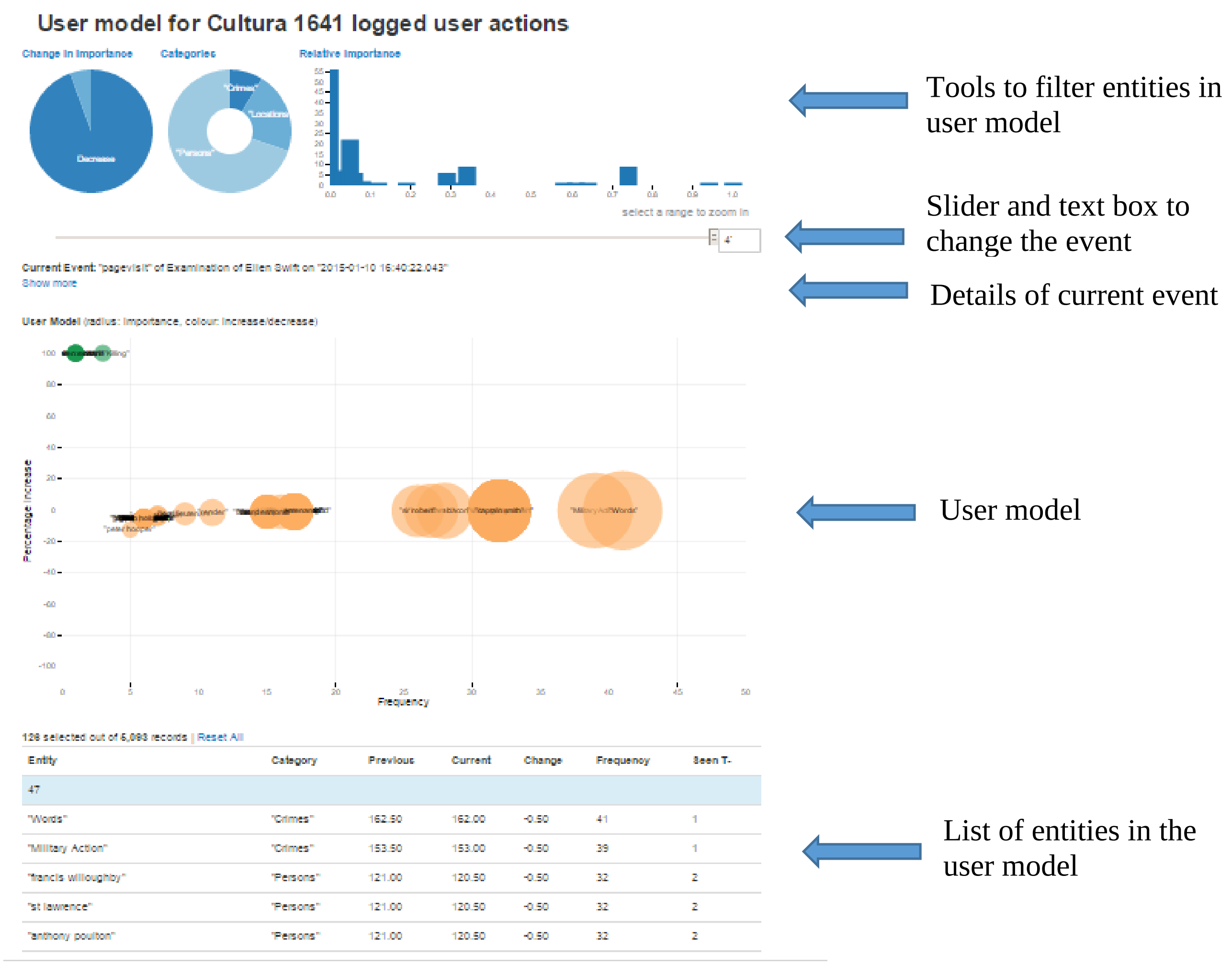
[WAMP] WampServer: Apache, PHP, MySQL on Windows (WAMP). Available online at http://wampserver.com/en/.

Appendices

Appendix A – Task Based Evaluation

Explore and interact with the user model through the visualizations

This data is generated from research undertaken by an individual into depositions from the 1641 Irish Rebellion. It is from users who are not individually identifiable and are no longer members of the college community. You are going to look at a user model illustrating their interests based on their actions and the depositions they performed the actions on.


User model for Cultura 1641 logged user actions
Change in Importance
Categories
Relative Importance
Decrease
"Crimes"
"Locations"
"Persons"
select a range to zoom in
User Model (radius: Importance, colour: Increase/decrease)
Percentage Increase
Frequency
Tools to filter entities in user model
Slider and text box to change the event
Details of current event
User model
List of entities in the user model

Current Event: "pagevisit" of Examination of Ellen Swift on "2015-01-10 16:40:22.043"
Show more

128 selected out of 5,093 records | Reset All

Entity	Category	Previous	Current	Change	Frequency	Seen T-
47						
"Words"	"Crimes"	162.50	162.00	-0.50	41	1
"Military Action"	"Crimes"	153.50	153.00	-0.50	39	1
"francis willoughby"	"Persons"	121.00	120.50	-0.50	32	2
"st lawrence"	"Persons"	121.00	120.50	-0.50	32	2
"anthony poulton"	"Persons"	121.00	120.50	-0.50	32	2

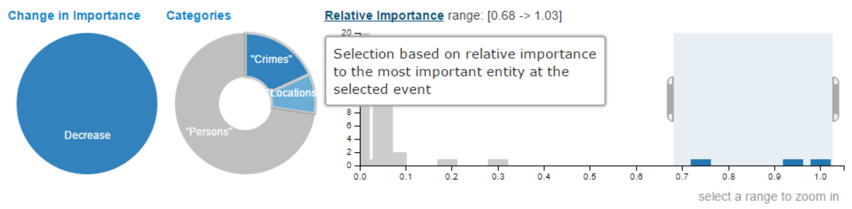
Figure 1: User model interface

The interface will appear as seen in Figure 1. For the purpose of this experiment, an entity is an object – a person, a place or a crime – that appears in the depositions. When a user performs an action on a deposition, the entities that appear in that deposition will be used to build the user model, from which we can gain insight into the user's interests.

The charts in Figure 2 can be used to reduce the number of entities displayed in the user model by concentrating on certain properties.

- Change in Importance – this pie chart allows you to select only entities that have increased in the current event, that is entities related to the current deposition, or only entities that have decreased in the current event, that is entities not related to the current deposition that still appear in the user model.
- Categories – Entities for these depositions are classified as one of the following: Crimes, Locations, Persons. The Categories chart allows you to specify which categories appear in the user model.
- Relative Importance – Relative importance is the importance of an entity relative to that of the most important entity of the selected event. A range can be selected and adjusted by dragging the edges of the selection. The selection can be removed by clicking in the white area of the graph.

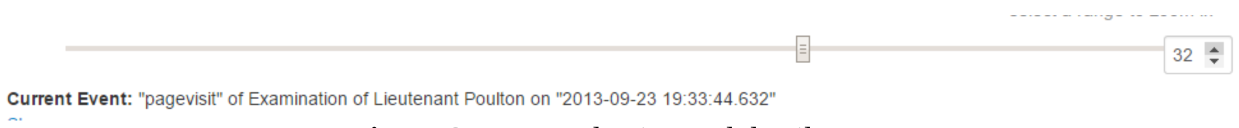
For a description of the purpose of the graph, you can hover over their title at any time.


Change in Importance
Categories
Relative Importance range: [0.68 -> 1.03]
"Crimes"
Locations
"Persons"
Decrease
select a range to zoom in

Selection based on relative importance to the most important entity at the selected event

Figure 2: Tools to filter entities in user model

You can move through the events using the slider, typing the required event number in the text box or moving the arrows up and down on the right of the text box. For each event, the user action, the title of the deposition and the date are printed below the slider. This can be seen in figure 3.



Current Event: "pagevisit" of Examination of Lieutenant Poulton on "2013-09-23 19:33:44.632"

Figure 3: Event selection and details

The user model can be seen in figure 4.

- X-axis: The x-axis shows the frequency, the number of times an entity has contributed to the user model. Each time an entity is related to the event, it moves further to the right. Entities related to an event that are new to the user model will appear along the y-axis.
- Y-axis: The y-axis shows the percentage increase in importance of entities from the previous user model.
- Radius: The larger the bubble, the more important the entity. Each time an entity is related to an event, it increases in size. Entities in the user model at time i which are not related to event i+1 will decrease in size at time i+1 demonstrating a decrease in importance.
- Colour: Colour is determined by the percentage change in importance. The greener the entity, the more positive the percentage change. The redder the entity, the more negative the percentage change.
- Label: A label will appear for an entity if you hover over it with details of its current importance, change in importance and frequency.

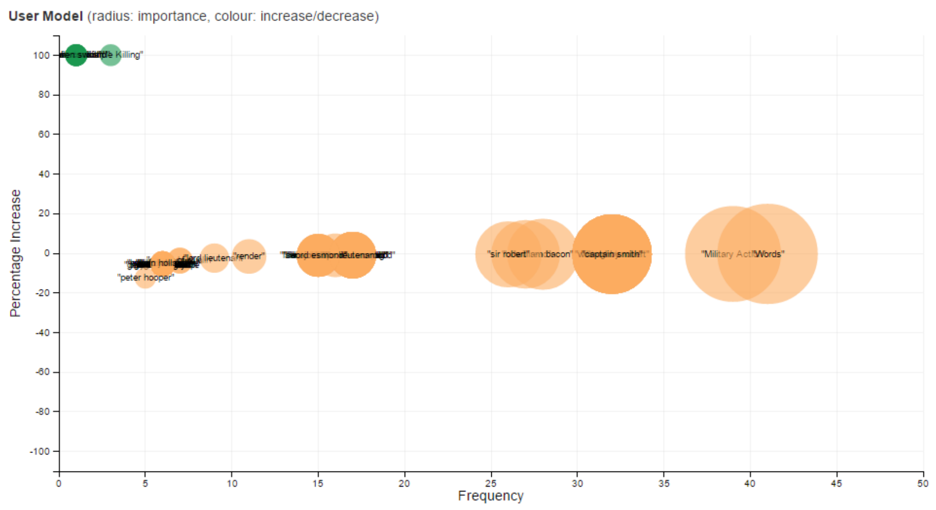
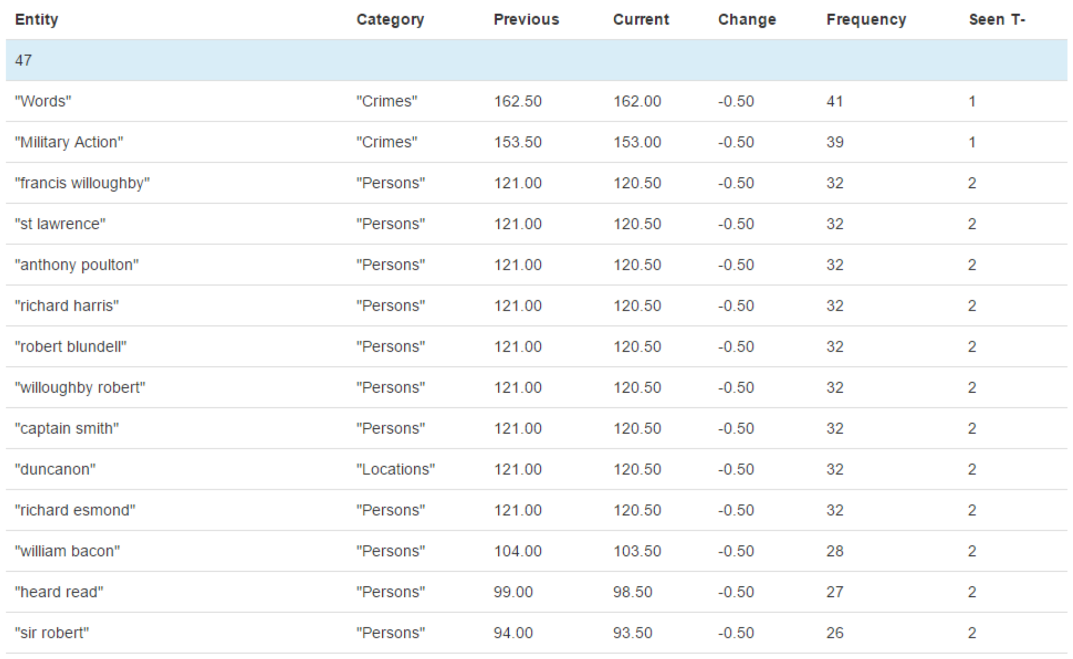


Figure 4: User model

Figure 5 shows an extract from the list of entities displayed in the user model. When the user model is filtered, the list only shows entities in the filtered user model. The name of the entity, the category it is classified as, the previous, current and change in importance, the number of times it has increased in importance in the user model and the number of events since it last increased in importance are listed for each entity.



Entity	Category	Previous	Current	Change	Frequency	Seen T-
47						
"Words"	"Crimes"	162.50	162.00	-0.50	41	1
"Military Action"	"Crimes"	153.50	153.00	-0.50	39	1
"francis willoughby"	"Persons"	121.00	120.50	-0.50	32	2
"st lawrence"	"Persons"	121.00	120.50	-0.50	32	2
"anthony poulton"	"Persons"	121.00	120.50	-0.50	32	2
"richard harris"	"Persons"	121.00	120.50	-0.50	32	2
"robert blundell"	"Persons"	121.00	120.50	-0.50	32	2
"willoughby robert"	"Persons"	121.00	120.50	-0.50	32	2
"captain smith"	"Persons"	121.00	120.50	-0.50	32	2
"duncanon"	"Locations"	121.00	120.50	-0.50	32	2
"richard esmond"	"Persons"	121.00	120.50	-0.50	32	2
"william bacon"	"Persons"	104.00	103.50	-0.50	28	2
"heard read"	"Persons"	99.00	98.50	-0.50	27	2
"sir robert"	"Persons"	94.00	93.50	-0.50	26	2

Figure 5: List of entities in the user model

Tasks

Please Note: Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions are responded to.

1. At event 46:
   a. What action was undertaken by the user?
   b. What does the system perceive to be the top entity of interest?
   c. What does the system perceive to be the top 3 Crimes of interest?
   d. What does the system perceive to be the top 4 Locations of interest?
   e. Outside of these 4 Locations, how many Locations increased in importance?
2. Staying on event 46, adjust the user model to look at the entities whose importance is between 0.3 and 0.7 relative to that of the most important entity.
   a. Are there any Persons in this range?
   b. If so, what is the most important Person of interest?
   c. Are there any Locations in this range?
3. Look at the entities whose importance is between 0.3 and 0.7 relative to that of the most important entity at event 29
   a. Are there any Persons in this range?
   b. If so, is the Person from question 2b in this range?
   c. If so, is this Person still the most important Person of interest?
   d. Are there any Locations in this range?
   e. If so, list the Locations in order of importance
4. How do the user models for events 29 and 46 compare, in particular looking at the Persons and Locations from questions 2 and 3?
5. Starting at event 10 and progressing in 5 event intervals, what do you notice about the user model?
6. In event 45, how many times does an entity containing the name Robert occur?
7. Can you select the entity containing Robert with the lowest weight?
8. During the exploration, the user performs the following actions: Pagevisit, Annotation, Bookmark. Which is most important task users perform when it comes to increasing their interest in things?

Appendix B – Post Questionnaire

Please Note: Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions are responded to.

Change in Performance Visualisation

1. The Change in Performance visualisation was user friendly

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

2. The Change in Performance visualization was not useful to filter the user model

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

3. The title of the Change in Performance visualization was self-explanatory

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

4. The labels of the Change in Performance visualization were not easy to understand.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

Categories Visualisation

5. The Categories visualisation was not user friendly

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

6. The Categories visualisation was useful to filter the user model.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

7. The title of the Categories visualization was not self-explanatory

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

8. The labels of the Categories visualization were easy to understand.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

Please Note: Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions are responded to.

Relative Importance Visualisation

9. The Relative Importance visualisation was user friendly

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

10. The Relative Importance visualisation was not useful to filter the user model.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

11. The title of the Relative Importance visualization was self-explanatory

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

12. The labels of the Relative Importance visualization were not easy to understand.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

Event Selection and Details

13. The methods of event selection were user friendly

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

14. It was not easy to select a specific event.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

15. The event details were easy to understand

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

Please Note: Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions are responded to.”

User Model Visualisation

16. The User Model visualisation was not user friendly

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

17. The tools provided were useful to filter the user model.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

18. The labels of the user model visualization were not self-explanatory

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

19. The importance of the entities was evident from the user model visualization.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

Detailed list of entities

20. It was not useful to have entities for the current user model with filters applied listed.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

21. The column titles of the table were easy to understand.

☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly Disagree

Comments ____

Appendix C – Usability Questions

Please Note: Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions are responded to.

	Strongly Agree				Strongly Disagree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5