Optimize the interaction by knowledge graph

Xiang Mao

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Intelligent Systems)

Supervisor: Declan O'Sullivan

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Xiang Mao

August 20, 2022

Permission to Lend and/or Copy

Optimize the interaction by knowledge graph

Xiang Mao, Master of Science in Computer Science

University of Dublin, Trinity College, 2022

Supervisor: Declan O'Sullivan

There is a basic need for information, and one of the most common ways to obtain information today is through the internet. One of the most common ways of accessing the information on the web is through search engines, and one of the most common ways that search engines display information is through text listings. However, knowledge graphs perform better than text lists for this type of information, and as people access more and more information on the web, text lists are becoming more and more difficult to cope with the complexity of information needs. However, knowledge graphs are not perfect and have some problems, so this paper will design an approach to optimise user interactions with knowledge graphs.

Acknowledgments

Thank you very much, Professor O'Sullivan. You have helped me a lot. Thank you for your help.

Thank you very much, Professor Wade, for watching my demo.

Thank you very much, Professor Stephens. for talking to me and for all the help you have given me.

Whoever is reading this and is ready to start your dissertation, I want you to know that it is not whether you know anything, but whether you know the research that is important. You will find that it is not a particular area of knowledge that you are learning about, but a way of gaining knowledge in any area.

Xiang Mao

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Obtaining information is a fundamental activity for people (52). Although there are many ways of obtaining information, for example: face-to-face communication (70), telephone interviews (78; 25), libraries (64; 24), but increasingly popular access to information has been replaced by the internet (29). The increase in information on the web has led to information overload (44), which has placed an increasing burden on users using the internet to access information, and this has affected the user experience of using the internet to access information (32; 10). Although search engines are an important tool for users to access information on the web (46), However, there is a growing awareness in the search field that is interacting with search results by arranging them in a list on a page is no longer sufficient to cope with the complexity of information needs (2).

Search tasks and information exploration are better performed when different information nodes are connected by their relationships through knowledge graphs (65; 1; 41; 66; 3; 87; 92)

As a result, knowledge graphs can be much easier to extend content through relational reasoning and graphical structuring to obtain more valuable information, which is a great advantage compared to the text list layout of textual information.

While the Knowledge Graph has improved text list information presentation in some ways, the Knowledge Graph still suffers from some problems such as: Too Many Nodes. As the visualization community is well aware, displaying massive graphs can create viewability and usability issues. (41)

However, Information visualization (InfoVis) can amplify and augment the cognition of users (38). (9) and (37) provide a broad definition of InfoVis: Information visualisation (InfoVis) is the use of interactive visual interfaces to amplify human cognitive and abstract data communication. InfoVis can help users analyse, explore and understand data through progressive, iterative visual exploration (74).

InfoVis enhances the user experience by improving the user's perception of the content. For example: Some of the key ways in which vision can amplify perception. (9). This section is discussed in 2.2.1.

Therefore, this Dissertation will improve the interaction experience and user satisfaction of knowledge graphs by improving these essential elements. It will also evaluate the effectiveness of the user experience of the visualised and improved knowledge graph interactions.

1.2 Aims

- Design an approach to optimise user interactions with knowledge graphs
- Evaluate the approach with user interface satisfaction as a measure of interaction effectiveness

1.3 Research Question

1. How can an optimised approach using Knowledge Graphs improve the state of user interaction?
2. Are interaction effects influenced by the method of optimisation?

1.4 Research Objectives

In order to achieve the objectives of this study, it is necessary to examine the following objectives.

- State-of-the-art review of literature on interaction design, information visualisation
- To Select interaction approaches for implementation
- To Select technologies for implementation
- Evaluation and analysis of interaction effects

1.5 Expect Contributions

This study is expected to contribute to the field of interaction with knowledge graphs and to the field of information visualisation. It will also serve as a test case for using user interface satisfaction as a metric in these areas.

1.6 Outline of Dissertation

In this first chapter, the motivation and objectives of this research are presented. In the second chapter, the current state of knowledge graph interaction research is presented, with a state-of-the-art review of the literature on interaction design and information visualisation and a description of the existing problems. In Chapter 3, visualisation and interaction design will be discussed. In Chapter 4, the implementation process of the knowledge graph interaction approach designed in the previous chapter will be discussed, the techniques used, the reasons for their use, and the problems encountered in the implementation process and their solutions, including system architecture design, front-end implementation, back-end implementation and the development process of the graph database. In Chapter 5, the results of the study are evaluated. Finally, in Chapter 6, conclusions will be drawn from the research, and future work will be discussed.

Chapter 2

State of the Art

This chapter provides an overview of the background to knowledge graph interaction design and describes concepts related to information visualisation, interaction design and knowledge graphs. It will also discuss the literature on visualisation, interaction design and evaluation, and the literature on knowledge graph implementation techniques.

2.1 The concept of knowledge graphs

2.1.1 RDF

Resource Description Framework (RDF) can be used to represent metadata of other web resources, e.g. XML files (16). RDF is a graphical data model that can be used to model data using RDF(S) and to build graph databases by studying it and other related technologies(23).

2.1.2 Knowledge Graph

The knowledge graph can be described in different ways (4; 20; 58; 16; 23), and its precise definition does not seem to There is no widespread agreement, and the main focus of this Dissertation is on interaction and visualization 2.2. Therefore, an attempt is made to describe and organize the knowledge graph elements from a visualization perspective.

2.1.3 Node, Relationship, Predicative relationship

In (20)'s study, it is stated that "knowledge graphs (i) primarily describe real-world entities and their interrelationships, organised as graphs, (ii) define possible categories and relationships of entities in schemas, (iii) allow for possible interrelationships between arbitrary entities, and (iv) cover a variety of topic areas." "A knowledge graph is a large network of entities, their semantic types, attributes and relationships between entities." Knowledge graphs need to describe different entities and relationships and organise them graphically (58). So the knowledge graph needs to have two graphical features for visualisation: (1) a graph representing the entities; and (2) a graph representing the relationship between these two entities. In (41), such entities are described as nodes, and the relationship between the two entities in time is described as a "relation", which is also referred to as a "predicate relation" in (23).

2.1.4 Ontology, Entity

In (4)'s study, which explores the concepts of "Ontology", "Knowledge Graph", and "Knowledge base", the authors state that there is no difference between an ontology and a knowledge base and that a knowledge graph acquires and integrates information in a knowledge base. Therefore, the knowledge graph also captures and integrates information in Ontology, while the knowledge graph describes nodes and relations 2.1.3. So Ontology has some correspondence with nodes. Therefore, in this study, since the focus of the research is on interaction and visualization, in order to use the concepts accurately and to distinguish the figurative part of visualization from the abstract part of model description and construction, "node" will be used in the visual interaction design part 3 to describe the node and highlight its graphical features; in the system implementation 4 The system implementation 4 will be described by "Ontology" and "Entity", highlighting its characteristics as a data carrier.

2.1.5 Ontology modeling

In (73), the author applies several methods for building an ontology, including modelling the ontology through 5 steps, step 1: define a use case or use case scope; step 2: collect capability problems while viewing possible data sources and scoping the problem, i.e. deciding what should be modelled now and what should be left for possible later expansion; Step 3: Identify key concepts from data and use cases and decide which should be used the pattern, then build a set of models; Step 4: Put modules together and add axioms involving multiple modules; Step 5: Create an OWL file.

These five steps seem to be sufficient for many different datasets, but the application of this approach requires that the dataset is defined, that there are specific problems, and that possible future scenarios need to be conceived in advance. Does the modelling in this way provide any insight into how many future scenarios are unanticipated? Is there a way to think of all possible future scenarios? Or is there a way to measure the percentage of the future that has not been encountered? The authors do not provide answers to these questions. As this Dissertation focuses on optimising interactions with knowledge graphs rather than optimising for a particular dataset, some changes to these steps may be required when building ontologies in the way the authors do. Ontology modeling is discussed in 4.3.1.

2.1.6 R2RML

The RDB to RDF Mapping Language (R2RML) is described in detail in (77), a detailed tutorial containing a definition of R2RML and how to use it. These will be used in 4.3.1.

In (15), the author examines R2RML-F, an extension to R2RML that allows relational data, such as data stored in a '.CSV' file, to be uplifted to an RDF dataset, which ends up as a '.ttl' file, which can be uploaded to a triple store. The content of the graph data will be discussed and implemented in 4.3.1.

2.2 Information visualisation

2.2.1 Concept

在 (9) 中, the author define information visualization(InfoVis) broadly: infovis is the use of computers to interactively amplify cognition, using visual representations, Some of the key ways in which vision can amplify perception include:

1. Increasing memory and processing resources available
2. Reducing search for information
3. Enhancing the recognition of patterns
4. Enabling perceptual inference operations
5. Using perceptual attention mechanisms for monitoring
6. Encoding information in a manipulable medium

Some of these methods are practical for knowledge graphs, for example, increasing the available memory and processing resources. Since too many nodes in a knowledge graph can directly affect the user's recognition of content 2.3.2, in this study these methods will be used as an important basis for optimising interactions and theoretical foundations, which will be described in 3.13.33.4.

2.2.2 InfoVis

InfoVis is a field of study that aims to help users explore, understand and analyse data through progressive, iterative visual exploration (74). So the aim of the InfoVis field is: to explore, understand and analyse data. This means that the key methods visualised in 2.2.1 also have the ability to help users explore, understand and analyse data.

InfoVis is being used in various data analysis applications, and in (8) applies visualisation design to track the spread of information in social media in real-time. Their design investigates three critical features of the communication process in social media: temporal trends, social, spatial scope and community reactions to topics of interest. In the study, the authors tag different nodes in the knowledge graph with different colours to enhance the recognition of different content, allowing content to be easily identified by enhancing the comparison of visual patterns, as explored in 3.3, a similar approach to improving visual interaction by enhancing the recognition of patterns.

In (13), visualisation and topic mining techniques are combined to analyse the various evolving patterns that emerge across multiple topics to investigate how topics evolve in text data. The study also includes two visual interaction methods: graphical symbols representing information or relationships and click selection events for content on a page. The graphical symbols in this study, although they can be used for visualisation tasks, are defined by the author and are not directly understood by the user, but require a simple learning process to know the meaning of the symbols, e.g. the author uses solid circles for sources and hollow circles for sinks, but is there a natural connection between these two symbols and what they represent, This is not explained. In contrast, when used in place of a directional message, symbols such as the arrow symbol 2.3.2 allow the user to reason intuitively. Intuitive reasoning about symbols 3.4.2, and its application to visual interactions, will be explored in later chapters.

In the (51) study, an interactive visualisation application was developed and evaluated to allow science museum visitors to explore content by viewing and interacting with scientific data. Using visualisation design, the system was able to engage visitors on a personal level, and visitors often lacked the background to interpret scientific data visualisations and spent minimal time on individual museum exhibits. The use of a layered approach to information access for visitors to display information allows visitors to focus on simpler content at a time, rather than displaying large amounts of data in its entirety, and the authors argue that open exploration can be difficult for novices who are quickly overwhelmed by the sheer volume of data types and representations. Therefore, when there is a large amount of information on a page, it is possible to hide some of it and display it when the user needs it, thus making it less difficult to explore the information.

This approach is used in super-nodes 3.1.2 and non-modal dialogues 3.1.1.

2.2.3 Visualisation and awareness

In (61), the authors discuss research on information visualization and human cognition, describing the basic idea of visual cognition as a mechanism of a visual perception whose output is a symbolic representation or 'structural description' of a scene, i.e. aspects of vision such as position, colour, texture, size, shape, constituent elements and their spatial relationships between them, etc. These visual factors all influence the user's perception of the content. It is also a fact about human cognition that humans generally prefer to process quantitative information in the form of graphs, such as line diagrams, bar charts, pie charts, Venn diagrams, flow charts, tree structures, networks of nodes, etc. (11; 14; 71; 86).

In (Larkin and Simon) the authors argues that graphic descriptions are better suited as cognitive aids than more conceptual representations because of their ability to visualize abstract relationships. It also investigates the role of graphics for information processing tasks, using semiotic and psychologically based theories of graphic design and interpretation. In it, the effectiveness of visualization design is investigated through two approaches. These are using visual symbols as a substitute for logical reasoning and reducing the user's search for the required information. Through experimentation, the authors found a statistically significant two-fold difference in user response time to graphics between systems designed using these two methods and those designed without them. In this study, although the difference in time was significant, the length of time could be explained by the different efficiency of the different methods. However, the impact of the change in efficiency on user satisfaction and user experience was not evaluated in this study, so it can only be assumed that the change in efficiency here could potentially have an impact on user satisfaction, but it is not known whether the extent of the impact was also two times greater.

In (55), the authors provided empirically-supported guidelines for increasing the efficiency of graphic presentations as a means of quickly conveying information. These include: (1) allowing users to substitute perceptual inferences for more demanding logical inferences; (2) reducing search for needed information. This is captured by a belief that distinguishing the "steps" taken to process a graph could predict response times (50).

These studies all seem to elaborate on these methods, but there is a lack of a more direct and explicit quantitative way of evaluating, for example, QUIS5.1.1, and then numerically presenting the level of user satisfaction with the interface.

More work (43) places more focus on interaction rather than the representation, but with the same ‘cognitive cost’ motivation.

In (33) this research, the authors explored the cognitive load, cognitive load represents mental effort, or the amount of cognitive capacity that is allocated to accommodate the demands imposed by a task. When memory demand is consistently lower than maximum memory capacity, task performance (e.g., accuracy, response time, perceived effort) is likely to be good. assuming the load is not so low as to lead to boredom. When memory demand reaches maximum capacity, performance plummets. Design approaches to reduce cognitive load are discussed in 3.

Thus, how cognition can be enhanced through images can be understood as allowing users to intuitively reason through the meanings that images have, as discussed in 3.4.2, by perceiving content through the meanings contained in symbolic images, or by making the content that users need more visible through images, or by excluding unwanted content, as discussed in 3.3.2 discusses categorizing large amounts of information by colour so that users can exclude unwanted nodes by colour, or find a desired class of nodes by colour selection.

2.2.4 Colour

The use of colour is a fundamental and essential technique for visualising information and has a long history of (84). In the (67) study, they examined the correlation between colour and aesthetic preference, in which they found a general trend: high-contrast images and those images whose colours are mostly Higherranked images tend to be highly saturated and colourful but have low contrast.

Through the research of the above authors, we can understand the importance of colour and at which interval the colour will have a better display experience. However, in their research, colourblind people are ignored. When colourblind people use the system, there may be recognition barriers, and in the (88) study, colours have been developed that can be used by colourblind people. However, these colours do not necessarily fit everyone’s aesthetic.

2.3 Human-computer interaction

2.3.1 The concept of human-computer interaction

Concept of Human Computer Interaction (HCI) was automatically represented with the emerging of computer, or more generally machines itself. HUMAN COMPUTER INTERACTION is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them (75).

2.3.2 Interaction in visualisation

Interaction has a critical role in visualisation, enhancing visualisation and helping users understand data better. (90) presents a comprehensive survey to investigate the role of interaction techniques in InfoVis. In this section, the study of interaction in visualisation is examined. The interactions described in this section refer to the application of interactions to visualisation. The main focus of this section is on interactions with knowledge graphs.

Complexity due to too many nodes

Although knowledge graphs allow information to be explored through the relationships between nodes, they also show a better experience in exploratory search than arranging information in a list (65). As the visualisation community is well aware, there are visibility and usability issues when displaying massive graphs, the content of which is difficult to understand and analyse without additional manipulation (31). In a study by (41), the visibility problems encountered when there are thousands of nodes in a knowledge graph were investigated, and attempts were made to solve the visibility problems encountered when there are a large number of nodes in the graph by using super-nodes, which combine thousands of nodes into a single node.

Indented Tree and Graph

(21) defines visual scalability as the ability of a visualisation tool to display large data sets effectively. (91) conducted a study on complexity and graph visualisation, and their analysis indicates that the floor effect of interaction lies at the boundary of nodes equal to 4 and edges equal to. The knowledge graph degrades the user experience when the mesh relationship becomes complex, as different data may have different depths. In (26)'s study, mesh knowledge graphs and tree knowledge graphs were investigated. This study compared the effectiveness of tree and mesh knowledge graphs in completing tasks. The validity results show that indented trees have better validity when the number of nodes on a page is high and the structure of the relationships becomes complex. The researchers concluded that indented tree visualisations are more suitable for list checking activities, while mesh diagram visualisations are more suitable for overviews. The study also pointed out the disadvantages of indented trees: with fixed screen size, indented trees can be challenging to display if the nodes in the tree have a large number of descendants

because when a large number of descendants of a node are displayed on the screen at the same time, they take up the display space of other nodes.

The use of indented trees therefore makes it easier to explore content, but has the potential to cause display problems when there are too many nodes, which can affect the display of other content. This is attempted in 3.1.2 and 3.1.1.

Comparing the content of large tree structures is a difficult task, and some of the current visualisation techniques can partially support this, but they are designed primarily for browsing. In (56) they propose a method that can be used to compare the detailed structure of two trees with each other. The method is designed to visualise (1) the different tree structures side by side, (2) ensure the visibility of the same root nodes of the different tree structures, and (3) make the different trees display different highlight colours. This tool is also useful in many other application areas where tree comparison is required, from network management to call graph optimisation to genealogy. The use of trees for visualising large amounts of data and comparing information has also been investigated by (83), in which they use the idea that trees can hierarchise content by designing the tree as a rectangular structure, wrapping child rectangles with parent rectangles to form a tree hierarchy. Feedback from users in the study suggests that the ability of the tree diagram to layer information helps users to explore information.

Thus, by retaining the results of the previous task and applying them to the subsequent task for comparison, the user can concentrate on the next task without remembering the previous results. This way of reducing the user's working memory helps to improve the user experience. However, in (56), the problem of too many nodes is not discussed in detail and is not addressed in detail, but only in terms of the overall structure of the comparison task.

Non-modal dialogue

A 'non-modal', or modeless, the dialogue box is a dialogue box that does not prevent or interrupt the user from interacting with other content on the system. In the (72) study, a 'non-modal' dialogue box was compared with a 'modal' dialogue box (also known as a 'non-modal' alert and a 'modal' alert), which was more effective in alerting the user in the experimental alert scenario but was more intrusive to the user. When the content of a modal dialogue is not important for the user to immediately interrupt interaction with the rest of the system, it is unnecessarily disruptive to the user. Non-modal dialogues, on the other hand, can be popped up as windows without interrupting the user's interaction with the rest of the system, allowing the user to choose whether or not to interact with the non-modal dialogue. This way, the user is less disturbed, and the user's perception of the importance of the content in the non-modal dialogue box is reduced. The design of non-modal dialogues will be discussed in 3.1.1.

Visual comparison

(82) investigates whether it is possible to provide more general support in terms of interaction when performing comparison-type tasks. A new concept of interaction is proposed by the behaviour of people comparing information on paper in the real world. This approach, consistent with real-world interaction, can support users in (1) interactively specifying the graphical information to be compared, (2) arranging this information flexibly on the screen, and (3) performing actual comparisons with side-by-side and overlapping arrangements of graphical information. In this study they conducted a qualitative user study, and the feedback obtained indicated that the experimental interaction techniques mimicked natural behaviour well, could be learned quickly, and could be easily applied to visual comparison tasks.

However, this study has only conducted qualitative experiments in a few fixed scenarios, and the impact of other interactions, such as clicking on the screen, dragging, etc., on the comparison task has not been determined. It is also challenging to maintain validity between a large number of dense comparisons, e.g. when comparing two knowledge graphs with a large number of nodes, which may be difficult without optimisation.

(30) describes an approach to exploration and discovery in large datasets. In this study, researchers grouped historical climatological data from around the world and used different visualisations to aid users' understanding of the data.

In (19) the authors investigated the semantic and structural aspects of the data. In their study, they used a way of displaying content by dividing the interface into three main areas, each showing a different type of node, and by collecting participants' comments on their use, they learned that this visualisation facilitated users' exploration of the data. In this study, the researchers applied the same design to different data to illustrate the versatility of the design across different information spaces.

The above research has revealed two ways of making visual comparisons: spatial comparisons(82; 19) and colour comparisons(30). Visual comparison can be applied to the design of 3.1.1 and 3.3.1.

Word-Picture

The study of visualisation and interaction in (9) shows that people do not read all the text word by word when reading content on a web page, but instead they understand the word-picture and that the more different the words are graphical, the easier it is for the user to read them.

However, if there is no more treatment of how words are displayed on a page, and they are simply scattered across the page, the problem of illegibility can still arise when the number of words is large enough (31). The use of word pictures is explored in 3.3.1.

Arrow symbol

In (85) we learn that arrow symbols can represent causal relationships. The arrow symbol is also universal, and readers of arrow symbols can recognise and interpret them almost intuitively (42). Arrow symbols also play an important role in visual communication, as they can be used to describe dynamic processes even in static diagrams.

As a result of these properties, the user who sees the arrow can intuitively determine that there is a causal relationship between the objects on either side of the arrow. However, the arrow symbol alone, without the addition of other visual interaction elements, will not be able to convey much more than an explanation of the existence of a relationship or pointing, e.g. when using the arrow symbol to describe a parent node with two different types of children, as in figure 2.1, when no other visual interaction method is used, the arrow symbols pointing to the two child nodes are identical, leading to a limit on the ability of the arrow symbols to represent the relationship between the nodes fully. This is explored in 3.4.2, which exploits and improves upon the already existing causal representations of arrow symbols to enhance them.

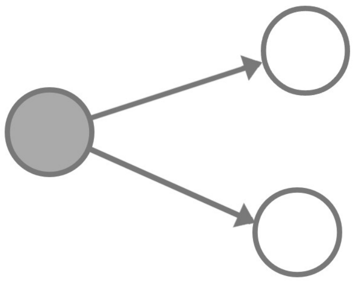


Figure 2.1: Relationships with missing type information

Navigation bar

In the (80) study, the navigation characteristics are described. Navigation can have a cascading structure, a single-level structure, or a multi-level structure and can be used to locate the current page within the system page structure or to jump to any level of the page quickly. It is, therefore, clear from this study that navigation can mark the current step in the interaction process and is used in 3.2.1 to assist the user in obtaining information about the path to a node. In the (81) study, the user uses a Navigation and Exploration Interfaces with a navigation bar with a node path, and the user has no problem finding relevant information during the navigation task. Although there are other ways of interacting in this interface than just displaying the path of information, it is not possible to exclude the positive or negative impact of the navigation bar on the interaction experience as it exists in conjunction with other methods of interaction between the user and the system.

In (48) and (89), basic interactions such as selection, filtering, swiping and highlighting are investigated. (48) builds an interactive, visual text analysis tool that helps users analyse large amounts of text. (89) Perform effective visual analysis of online customers' opinions. In this paper, they present an interactive visualisation system that can visually analyse a large number of online hotel customer reviews. Some of the interactive methods used in these studies, for example: Selection of systems that can be applied to this study.

2.3.3 Interaction without visualisation

As we can see from 2.3.2, visualisation is essentially a way of interacting with a machine. In addition to visualisation, some interactions can be performed without visualisation. This type of interaction without visualisation is not the focus of this Dissertation, so this section will only list three studies to illustrate that the concept of interaction is broader than visualisation.

(63) investigates interaction methods with in-vehicle devices through gestures without needing either a visual interface or mechanical buttons. (59) reviewed research on interacting with computer devices using only speech without visual content and concluded that, for drivers of vehicles, speech interaction generally improves driving performance over visual interaction, reduces subjective workload, and keeps the driver's eyes on the road and hands on the wheel. (17) and (68) discuss brain-machine interface (BMI) in their article. In this way, the user can interact with the computing device without needing a visual interface and hands, but only by "thinking" (i.e. the brain activity of the human subject). (47) proposes a way to control a wheelchair through BMI. This approach also does not require the involvement of a visual user interface or gestures.

In summary, human-computer interaction can be performed without visualisation.

2.4 Recursive

An informal definition of recursion is the process that a procedure undergoes when one of its steps involves calling the procedure itself. A recursive process is called a "recursive"(18). In computer science, recursion is a method of solving computational problems in which the solution depends on the solution of smaller instances of the same problem (27; 40). Recursion solves such recursive problems using functions that call themselves from their code. The method can be applied to many problems, and recursion is one of the core ideas of computer science (22). It is clear from these studies that recursion is similar in scalability to tree diagrams in that it can be extended continuously through a small instance.

2.5 Data

IMDB has a wealth of data on films, such as genres, actors, directors, writers and performers. There are approximately 3.6 million productions and over 7 million characters, including data on actors, production teams, fictional characters, biographies, plot summaries, trivia, and reviews (7). IMDB dataset resources provide a rich source of data for the testing and evaluation of many systems and have been utilised in many studies, for example (79; Melville et al.; Schickel and Faltings; 36) In this experiment, the data provided to the user for information exploration and verification of the system's interaction is the publicly available film data from the IMBD website for the past 40 years, which contains film data, actor data, director data and company data. Users will use the system provided in this study to explore the links between these different types of data.

Chapter 3

Design

The research in 2.3.2 shows that although knowledge graphs are better than lists of information when exploring information, and there are many ways to optimise interactions using knowledge graphs, However, these approaches still have drawbacks, such as the challenge of displaying a large number of nodes on a page (31; 41).

However, the user experience can be improved by improving the user's perception of the content2.2.3. Therefore, the design approach proposed in this study will improve these issues and optimise interactions in four key ways. These four key ways are:

increase available memory and processing resources: will be explored in 3.1.1 by using non-modal dialogues and supernodes to reduce the number of nodes and layers displayed at once, thereby reducing the working memory and cognitive load required by the user to complete the task, and thereby increasing the working memory available to the user to complete the task, thereby improving User perception of content ;

Reduce search for information: will be explored in 3.2.1 to assist users in identifying and searching for paths by way of navigation, reducing the need for users to actively search for information on the page and thus improving their perception of the content;

Enabling pattern recognition: will be explored in 3.3 to improve user perception of content by unifying the display pattern of node types, always in the same relative position and with the same colour relative to the node name, to enhance user recognition of patterns;

enabling perceptual inference operations: will be explored in 3.4.1, where parent and child nodes are represented by a horizontal change in relative position in the page through an indented tree, allowing users to infer which node is the parent and which node is the child through their perception of spatial position. This will be explored in 3.4.2, which allows the user to use the arrow symbols to determine whether the content of a message is related to a relationship and where the relationship leads.

3.1 Add available memory and processing resources

In this section, two improvements in interaction are discussed to increase the user's awareness of the content by increasing the memory and processing resources available. The two methods are non-modal dialogues and supernodes.

3.1.1 Non-modal dialog

In this section, we discuss the problems that arise when there is a lot of information in an interaction, what a non-modal dialogue is, how to use a non-modal dialogue to solve a problem, and how to design the non-modal dialogue. The non-modal dialogue designed in this section contains two key features: non-intrusive interaction and content-based resizing. This section concludes with a prototype of the non-modal dialogue box based on what has been discussed in this section, using the prototyping tool Axure(39).

Display problems with large amounts of information

When users are exploring information on a single screen, due to the limitations of screen size, it can be difficult to display when there is a large amount of information on a page, and it can also take more of the user's working memory and cognitive load, as explored in 2.2.3. Therefore, by reducing the amount of information displayed on a page while performing the same task, the difficulty of displaying large amounts of information on a page can be reduced, and the cognitive load on the user can be reduced, thus improving user experience and satisfaction, as applied in the (51) study, where unwanted information is temporarily hidden.

Non-intrusive interaction

As explored in (72), non-intrusive interaction is a method of interaction that does not interrupt the user's interaction with other parts of the system. Since a non-modal dialogue box is a pop-up window, it does not occupy the same plane as the current task but only floats to obscure or be obscured by an area, so the space within the non-modal dialogue box is an extension of the screen space, allowing more information to be displayed in the non-modal dialogue box.

This feature of non-modal dialogues allows them to expand the content that can be displayed on the screen while allowing the user to complete tasks smoothly (72).

Therefore, non-modal dialogues were chosen as one of the interaction optimisation methods in this study. As the number of nodes and the depth of the exploration level increase when using a tree diagram to explore information on a page, the height of the entire tree structure increases, taking up a lot of screen space, so non-modal dialogues can be used to expand the tree structure more, in a way that does not crowd the rest of the page or interfere with user interaction.

Variable size

As a window, a non-modal dialogue requires a basic rectangular area, so this dialogue is designed as a rectangle in its entirety. As the non-modal dialogue is presented as a tree, it is an indented tree with the ability to explore, and the overall height and width of the tree will change as the number of nodes and levels displayed at the same time are changed, and the tree is designed and explored in 3.4.1. Therefore the amount of information in the non-modal dialogue box will change as the exploration continues, i.e. the number and level of nodes displayed in the non-modal dialogue box are not fixed, so in order for the non-modal dialogue box to obscure as little visual space as possible, the area it obscures needs to change as the amount of information displayed in it changes, as the number and level of nodes in the non-modal dialogue box increases, it visually needs to When the number of nodes and levels in a non-modal dialogue increases, it visually needs to cover more space to display more nodes and levels, and when the number of nodes and levels in a non-modal dialogue decreases, it covers less area. 3.1 is a Non-modal Dialogue.

Prototyping

After the design discussion above in this section, a non-intrusive interaction is accomplished through Axure, as shown in 3.1

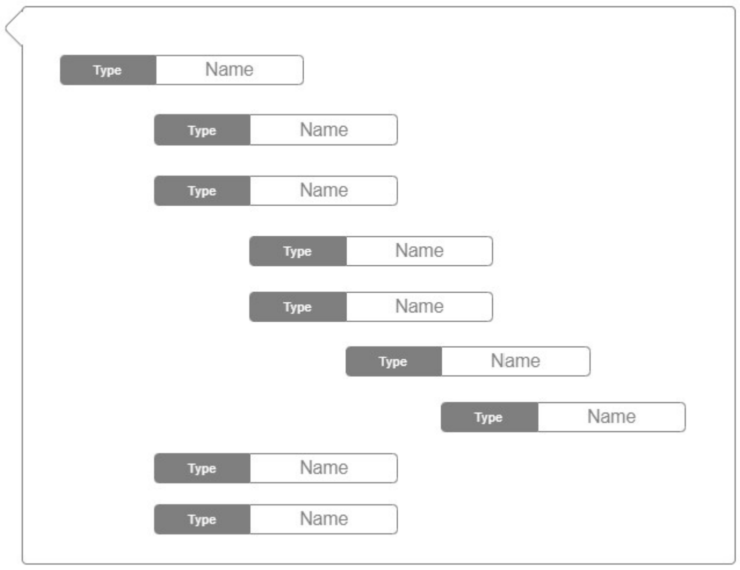


Figure 3.1: Prototype: Non-modal dialogue

3.1.2 Super node

The problems that arise when there is a lot of information on a page, such as crowding out space for other nodes, have been discussed in 3.1.1. A non-modal dialogue is proposed to optimise the interaction in terms of the amount of memory available to the user when interacting with the information on the page. The use of super-nodes in (41) is discussed in 2.3.2, but there are problems. In the study (51), interaction patterns are discussed by hiding parts of the content and showing it only when needed.

This section continues with a second way of increasing the memory or processing resources available to the user: the supernode. It also discusses the problem of supernodes losing information after merging, what a supernode is, how it can be used to solve the problem of large numbers of nodes in 3.1.1, and how it can be used to solve the problem of losing information about the number of merged nodes, and how it can be designed. The super node designed in this section contains two interactive features: the merged node and the uniform style.

Merge node

As the user processes information, short-term memory, i.e. working memory, is capable of remembering seven chunks(54). Therefore, when there are too many chunks, i.e. more than 7, it puts a burden on the user's memory. This affects the user's ability to complete tasks through such interaction. The number of child nodes should, therefore, not exceed 7. Therefore, the maximum number of nodes that can be displayed after hiding more than one node is 7. Therefore, in order to provide the user with as much information as possible at each level but with an as little negative impact on the user's memory as possible, the number of nodes to be displayed at each expansion is set to 7. All other nodes will be merged into one super node, and the super node will have a unique style that distinguishes it from other nodes. This section looks at the rules for displaying the number of super nodes and the reasons for using them, which are also part of the interaction. The style of super nodes is discussed in the next section 3.3.1.

Uniform style

The use and problems of supernodes are discussed in 2.3.2. When a supernode is merged with a large number of nodes, the display area is affected by the number of child nodes being merged, so when there are a large number of merged supernodes on a page, the different sizes of the different supernodes will result in different displays. Supernodes are also nodes, as discussed in 3.3.1 for 'Word-Picture', where enhanced recognition of patterns can increase user recognition of content, but here the super nodes have the opposite effect, i.e. different super nodes have different sizes, resulting in some super nodes being recognised by the user as a larger pattern and some smaller. Some super nodes are recognised by the user in a larger size pattern; others are smaller. As the difference in display mode affects the user's recognition of the content, it is possible to improve the display mode by making the different super nodes more similar in display mode, thus increasing the user's experience when recognising the supernodes.

The display elements that could be standardised are (1) size. The size of the node, which is affected by the number of children, is designed to be the same fixed size; (2) colour. Similar to the way 3.3.2 chose the colour for the "Word-Picture" node, colour is used in the current design to define different classes of nodes, so in order to make the user think less about the rules of colour and increase the cognitive load on the user, the colour of the node should be the same for all types of nodes. (3) Text. The (41) explored in 2.3.2 provides an example of a supernode application, but the merged supernode has no other information about these nodes, e.g. the number of nodes that have been merged is not shown on the node, so the user can only judge the number of nodes by their size, This approach can lead to errors because when the difference in the number of nodes between two supernodes is less than the difference provided by one pixel, the difference in size between the two nodes cannot be accurately compared and this information about the number is lost. Therefore, in the design of this study, the hidden information is retained by displaying textual information about the number of nodes in the node. Although the numbers in the text may be different, it is possible to standardise the content of the fixed descriptive text, as well as the style of the text; (4) Click interaction: In the (41) application, only the direct display of the supernode is provided, but no reasonable interaction is provided to switch the displayed content between the supernode and the normal node. In the design of this study, the simplest of the mouse-triggered events currently available in browsers, the click action, is applied to the supernode, as mentioned in 2.3.2. It is one of the seven types of interaction techniques. And as a node, the super node remains the same as a normal node in terms of the extended interaction pattern. This means that the user can expand or collapse a hidden child node by using a single super node.

3.2 Reduce search for information

The fundamental problem to be solved in this section is the same as in the previous subsections: too many nodes on a page degrade the user experience, but the difference in this section is how the user can interact with a large number of nodes to get the information they need in a way that is good for the user experience when the number of nodes displayed in the page can no longer be further reduced by other methods. This approach is explored in 3.2.1: navigation, an interaction method that does not reduce the number of nodes displayed on a page but increases the user experience by reducing the amount of information the user has to search for and increasing the user's awareness of the content.

3.2.1 Navigation

In this section, we discuss how navigation can reduce the number of information users has to search for. In this section, we discuss how navigation can reduce the need for users to search for information, the problems with the navigation bar, and how to solve them by designing the navigation with arrow symbols and timely updates.

The navigation designed in this section contains two key features: a display of relationships and timely updates. At the end of this section, the design of the navigation will be based on what has been discussed in this section, and the prototype will be designed using the prototype tool Axure.

Navigation-use

As discussed in 2.3.2, navigation can locate the location of the user's current interaction, so in an indented tree, navigation can provide timely feedback to the user on the hierarchy of information they are currently interested in, which can occur passively, i.e. the user does not need to actively search for and determine the hierarchy of relationships between nodes in the interface, but gets the results directly from the navigation bar. This reduces the need for users to search for information when they need to determine the content of a path or need to know the length of a path and reduces the amount of content that users have to remember when searching for a path themselves. This increases the user's knowledge of the content.

Navigation incomplete without relationship

As discussed in 2.3.2, (81) experimented with and evaluated a user interface with a navigation bar in their study, and although it worked relatively well, it still had problems; for example, the navigation was too homogeneous, only showing the nodes on the path from the start to the end, and not the connections The relationship between the two different nodes is not shown, and this unshown relationship may affect the user's perception of the content. For example, a node of the movie genre "The Mountain" can correspond to a node of the Person genre "Tom" through an "acted" relationship, while the same node of the movie genre "The Mountain" can still correspond to a node of the Person genre "Jack" through a "directed" relationship. Jack", these two people are not the same, and if the relationship is ignored in the navigation path, and the movie is the starting point, then no matter which of the above people is the endpoint, the user will not be able to determine what the connection between the movie and the corresponding two people is. The reason they are related is present in the relationship, but this key information is lost because the relationship is ignored.

Show relationship

The navigation in (81) is valuable, but if relationships are not included, the information in 3.2.1 is lost, so in this study, relationships need to be designed alongside the nodes displayed in the navigation. In order to reduce the cognitive load on the user, the predicate relationships displayed in the navigation will be the same as those in the indented tree, the design of which is discussed in detail in 3.4.2.

Update

The faster the user is given feedback on updates to the path information displayed in the navigation, the shorter the wait time and the smoother the interaction will be for the user. Because this is when the user first interactively focuses on a node, the expansion of the node takes up the mouse click event of the node, i.e. when the user clicks on the node, the clicked node expands and displays its children. Therefore the update of the path in the navigation will use the same mouse click event as the expansion of the node.

Prototyping

From the discussion of navigation in this section, it is clear that the navigation needs to have the following characteristics: (1) it should reflect what the user is interested in in the indented tree; (2) when a node in the indented tree is clicked, an interaction event is triggered to update the navigation path. (3) The navigation needs to show the complete information of the path, including the name of each relationship. (4) The elements in this navigation should have the lowest possible cognitive cost to the user, i.e. they should be as similar as possible to the relationships and nodes in the indented tree. The content designed through Axure is shown in Figure 3.2

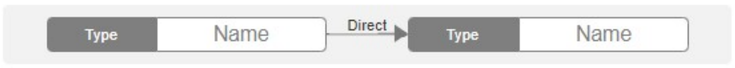


Figure 3.2: Prototype; Navigation

3.3 Enhance recognition of patterns

The fundamental problem to be solved in this section is that too many nodes in the page degrade the user experience, similar to the other sections 3.13.23.4, but this section differs in that the focus is not on changing the number of nodes. Instead, the focus is on optimising the interaction by changing the user's perception of the content, enhancing the user's recognition of patterns, and selecting the form of processing information that the user prefers, as discussed in 2.2.3. These affect the user's perception of content in terms of position, shape, size, colour, compositional elements, and spatial location.

In 3.3.1 we will explore how to enhance user recognition of node patterns and optimise interaction; in 3.3.2 we will explore how to choose colours for better recognition and user experience.

3.3.1 Word-Picture

In this section, we discuss ways to enhance user recognition of node patterns in the knowledge graph: rounded rectangles and combined structures.

Rounded rectangle

Since the user experience is better when the border shape of the icon is designed as an RS (Rounded Square), shape (49). Therefore, the basic border of each node in this experiment is designed with rounded corners.

Composite structure

When people read content on a web page, they do not read all the text word by word, but they understand the words graphically, and the more different the words are graphical, the easier it is for the user to read them. (9), so that by increasing the differences between ”Word Pictures”, it is possible to reduce the difficulty of accessing information and enhance the recognition of patterns by the user.

In the tree knowledge graph, the nodes will display the text related to the node content, In this study's knowledge graph, the node type is defined by ”rdfs:type”, and the node name is defined by ”xmd:name”, so every node of the same type, because their type is the same, will have the same type information when their type is displayed visually, i.e. every type of node in the system is connected to the same object by ”rdfs:type”. For example the movie ”The Mountain” and the movie ”River”, since they are both comedies, when they are displayed visually on the screen, the text of the movie names are different from each other, namely their names ”The The Mountain” and ”River”, but the text content of the genre is the same, both being ”Comedy”, and the film companies ”Columbia Pictures” and ”Universal Pictures”, both of which are displayed on the screen with different node names ”Columbia Pictures When the two nodes are displayed on the screen, they have different node names ”Columbia Pictures” and ”Universal Pictures”, but their node types are the same, both being ”Company”. Therefore, when the user interacts with the system, there are a large number of duplicate movie types, all of which are defined by ”rdfs:type”, and a large number of different ”xmd:name” nodes, i.e. the name attribute of the node. Therefore, when there are a large number of nodes on the page, grouping them by type will result in more nodes in each group than grouping them by name, so the user will be able to narrow down the selection of nodes by the type they belong to by eliminating the unwanted types first, which will eliminate more unwanted nodes faster than narrowing down the selection of nodes by unwanted names first.

According to (9), the visualisation of the Word Picture can be enhanced by enhancing the user's recognition of the patterns of the Word Picture. This means that the visualisation can be enhanced by enhancing the user's recognition of node types. To identify the type of node, the user needs to identify at least two pieces of information: (a) the relationship, i.e. what type of node the user is currently interested in, e.g. when four pieces of information are displayed on the page at the same time in the form of nodes: type ”Company”, type ”Movie”, company ”Columbia Pictures” and movie ”The Mountain”, Since they are separate from each other, the user needs to determine that the type of the ”Columbia Pictures” node is a ”Company” node and not a ”Movie” node, which requires additional judgement. (b) Identifying the content, i.e. the specific content of the current type of text, e.g. once the user has correctly identified the relationship and knows that ”Company” and ”Movie” are type nodes, the user needs to distinguish between the content displayed for ”Company” and ”Movie”.

In order for the node type, e.g. ”Movie”, and the entity node, e.g. ”The Mountain”, to be more relevant than any other node, the distance between these two different nodes can be minimised. In the case of overlapping nodes, The minimum distance is 0 pixels. By reducing the distance between the two nodes to 0 pixels, the two nodes are connected to each other as a closed whole, and a gap is formed at the top and bottom borders of the nodes due to the presence of rounded corners, as in the Figure: 3.3In order for the design in 3.3.1 to work, the whole needs to have the characteristics of a full rounded rectangle, so the gaps in the top and bottom borders need to be removed, as shown in the Figure: 3.3

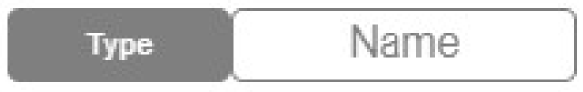


Figure 3.3: word picture: the top and bottom borders each form a gap

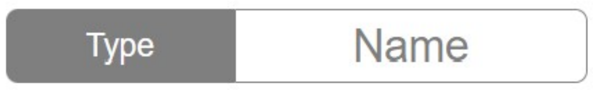


Figure 3.4: word picture: remove the notches that appear in the top and bottom borders

3.3.2 Colour

Since the meaning of different letters is determined by their structure, the display of a letter needs to conform to a specific structure. If this change is made after the visualisation, it will result in incorrect information. In the case of colours, which do not contain structural information, the colour is recognised as a whole at once, e.g. when recognising blue, only the colour ’blue’ is recognised at once in terms of colour, regardless of the shape of the container carrying the ’blue’, without loss of information about the ’blue’.

When displaying colour in a node, the maximum area of the colour can be the same as the area of the node, whereas the maximum area of the text in a node cannot be the same as the area of the node, as the text needs to have gaps in the graphics in order to retain structural information. It is also possible to change the font colour so that the colour features in the visualisation co-exist with the text features. Therefore, colour has a wider range of use in nodes and fewer structural constraints. Therefore, when improving the recognition of node types, colour can co-exist with text as a complementary recognition method, and colour will be more effective when it occupies more of the user's attention than text.

Therefore, the same type of node can be distinguished by a different colour. As discussed in 2.2.4, the choice of colours can be made in order to allow for effective content recognition even for the colour-impaired, so some already developed colours such as (88) can be used.

As highly saturated and colourful images are preferred by (67), the most saturated colours will be selected based on the (88) study, so this will be as balanced as possible between the average person and those who have difficulties with colour recognition. As shown in the figure 3.5.

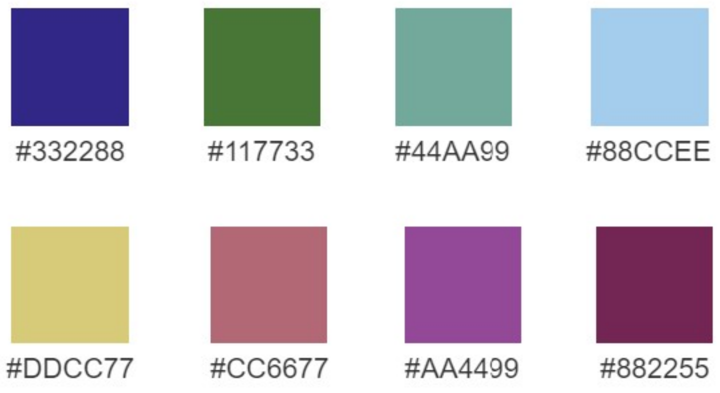


Figure 3.5: Color Card

3.4 Enable perceptual inference operations

The problem to be solved in this section is to make the user understand the relationship of information on the page with a low cognitive load. The approach used in this section is perceptual reasoning(9), as explored in 2.2.3, where space, location, shape and symbols affect the user's perception of content. The design of a perceptual model for indented trees is discussed in Section 1; the design of a perceptual model for predicate relations in indented trees is discussed in Section 2.

3.4.1 Indent tree

This section describes the reasons for choosing an indented tree, the basic structure of an indented tree and how perceptual reasoning is represented in an indented tree.

The 2.3.2 shows that the indented tree is still valid when there are a large number of nodes and complex relationships on the page and that it is suitable for list checking activities, i.e. when the nodes on the page are arranged in a certain list pattern on the page. On the other hand, the disadvantage of the indented tree is that it becomes difficult to display content when the nodes of the indented tree have a large number of children. The problems that arise with large numbers of children are discussed in detail and solved in **??**. This section will focus on the perceptual inference part of the necked tree. Based on these priorities, the indented tree is a valid solution.

Since spatial and positional relations will affect the user's perception, the same space and position will have the same ability to influence, so by designing similar spatial-positional relations for sibling nodes at the same level in the indented tree, the child nodes will produce the same progressive changes in spatial and positional relations, which will enable the user's perceptual reasoning in terms of spatial and positional relations. Since indented trees are good at list checking activities, changes in the number of tree structures can be extended vertically to mimic the characteristics of lists, and since the vertical space is already occupied by the increase in the number of nodes at the same level, the increase in the number of node levels can be done horizontally.

Thus indented trees are vertically aligned so that their siblings have similar spatial and positional characteristics, with the same indentation width on the left side, and the children, with increasing indentation as the hierarchy increases.

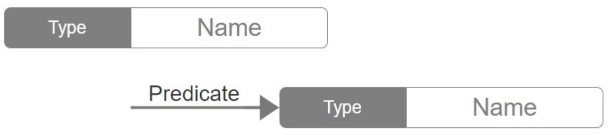


Figure 3.6: Indent Tree

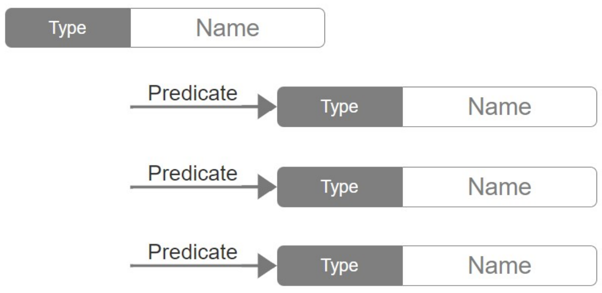


Figure 3.7: Indent Tree

3.4.2 Predicate

This section describes the visual element used to describe the relationship between two nodes in an indented tree: the predicate relationship, i.e. the relation.

As discussed in 2.3.2, the arrow symbol has the ability to initiate perceptual reasoning through intuition, but the arrow symbol still has limitations in expressing information: it fails when descriptive information other than that conveyed by the arrow's pointing is required. Since position in space affects the user's perception of content, descriptive content can be placed as text as close as possible to the arrow symbol as a complement.

In this study, since there are multiple predicates for connecting a node to another node, it is difficult to find a specific node by different relationships without distinguishing between nodes that are expanded in different ways in the knowledge graph, so it is necessary to distinguish between nodes that are expanded by different relationships in the knowledge graph.

Therefore, the distinction between different relations can be designed by pointing the arrow toward the child node (9) and showing the name of the predicate relation as far as possible from the arrow symbol. As in the diagram: 3.8

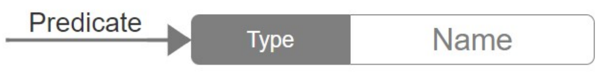


Figure 3.8: arrow symbol

Chapter 4

Implementation

In this section, the program development required to implement the interaction design content in 3 is explored. It also explores the reasons for the choice of technologies required to implement the system. In section 4.1 the system architecture and technology choices will be discussed as a whole; in section 4.2 the front-end technology choices and implementations will be discussed; in section 4.3 the back-end technologies and graph database technology choices and implementations will be discussed. The selection and implementation of back-end and graph database technologies will be discussed in section 4.3. In this study, the visual interaction is directly influenced by the front-end presentation, and therefore the front-end technology is the focus of this chapter.

4.1 System architecture design

In this chapter, the technologies needed to implement the system are discussed, the reasons for their selection and the overall architecture of the system. In the first section, the need to choose the right technology rather than multiple unsuitable technologies will be discussed; in the second section, a separate front-end and back-end development model will be discussed as a better approach for this study; and in the third section, the overall architecture of the system will be discussed as a result of the separate front-end and back-end development approach. As the front-end and back-end are developed as separate sub-projects, the front-end does not have direct access to the back-end data but needs to access the data through an API. (2) the API for exchanging data between the front-end and the back-end.

4.1.1 Choosing the right technology

There are many different technologies that are needed to achieve a function, and even if they are used for the same purpose, it is not necessary to use them all. (5), or native JavaScript(35)+HTML(62), but the use of React and jQuery together on the same page to achieve interactive effects may encounter Some limitations may be encountered. This limitation is not a physical impossibility or an inevitable risk that the application will not work when used together. This risk is unavoidable because of the underlying logic of how React and jQuery technologies work. The way React implements web interactions is that the developer describes in the code what needs to happen and what DOM(28) nodes need to be changed, and then React maintains asynchronously all the components that need to be displayed on the current page based on The advantage of this is that the developer no longer has to think about complex DOM nodes or the complex development logic that can arise when maintaining a large number of DOMs at the same time, and only has to develop the HTML as it needs to be displayed. Typescript(6), and can update the page only when the information on the page changes. When React and jQuery are mixed on the same page, React has no way of knowing what node is being updated by jQuery, so after jQuery has made a change to a node, React will not be able to update that node, and the program will error. For example, if you delete a button on a page and React tries to update that node as well, you will get an error. As the amount of code on a page increases, it is difficult to guarantee that this will never happen. Therefore, a greater number of choices of technology does not mean a better choice, but rather that the technology is appropriate.

4.1.2 Separate Backend Tech and Frontend Tech

Defining front-end and back-end technologies

Front-end technology is the technology used to implement the content used to interact with the user, while back-end technology is the technology used to support the interaction on the front-end. The front end and back end together pretty much determine most of the elements of an interactive experience, such as navigation, text, graphics etc.

Front-end development technologies, typically HTML, CSS, JavaScript libraries, jQuery, React, etc. The purpose of back-end or server-side code is usually to provide data management and interaction. In this study, the focus is on the implementation of the front-end interactive experience, so database-related content will be covered in the back-end implementation section.

Coupling and separation

Front-end and back-end coupling, i.e. mixing the front-end content with the back-end content. The advantage of having a mix of front and back ends is that development is fast, but the disadvantage is that updates and modifications can be difficult.

For example, when developing web pages in PHP, the front-end HTML, CSS, JavaScript code and PHP code can all be placed in the same '.php' file, and the back-end code can be used to display the results directly on the page with the code 'echo', saving development time by not having to exchange information between the front and back ends. For example, 4.1 is a simple example of mixing front and back ends together, where the front end and back end are coupled to create a simple function where a warning box pops up by clicking on the text 'Dissertation test'. In this code, a single piece of code in a page can be used to mix different technologies to achieve functionality for pages that have a single function, such as information display, and do not require the user to interact with the data. This makes development faster for simple information displays.

Listing 4.1: PHP code for show mixed

```
<div style="border: solid 1px #000;" onclick="alert('click')">
    <?php
        echo 'Dissertation test';
    ?>
<div>
```

However, because in the example above, a page has a mix of HTML, CSS, JavaScript and PHP, when changes are made to the page, it is necessary to iterate over which is the better way to make the changes. If there are other pages or functional modules in the system that use similar functionality, then the code has to be copied to other pages, and when the functionality needs to be modified, it is necessary to find which parts of all the pages refer to this code and then make the same changes in turn. There is a lot of duplication of effort in this whole process, and if there is a need for similar functionality in the system to be reused on other pages, the programming task will be repetitive and complex when these are reused on other pages, but each of the different pages needs to be modified. So, if there is functionality in the system that needs to be modified and reused, a coupled front-end and back-end approach is inefficient and cumbersome.

The biggest advantages of a separate front-end and back-end approach compared to a coupled approach, where there is only one developer, are three: reusable modules, ease of modification and ease of debugging.

When the same module is used on a page, the front-end and back-end are separated so that the same module can be made into a component or class and then used directly in a different module, so it is easy to modify this functionality because all the places where it is used refer to the same piece of code, so a direct change to this code will affect all the modules that refer to it. When debugging a program, if a problem is found, it is easier to locate the problem on the front or back end.

In this study, the system is an interactive visual system, not an information display page without data updates, and there are many locations where functionality is reused, e.g. nodes need to be displayed on a page and nodes need to be displayed in a pop-up exploration box. The system was developed using a separate front-end and back-end approach in order to allow for better visual interaction and to allow for quick modifications to the system to meet the needs of the study.

4.1.3 Overall system architecture

As a whole, the system consists of three parts: the front-end, the back-end and the graph database.

front-end : The front-end is only used for the implementation of interactions, and all visualisations are requested via the API (57) to the back-end. The front-end implementation will be discussed in 4.2.

back-end : The back end is the bridge between the front end and the graph data. When the back-end receives a query from the front-end, it processes the query string, identifies the content of the request, requests the corresponding data from the graph database, and finally returns the result to the front-end.

graph database: is used to store graph data and return the result to the back-end when it receives a query request from the back-end.

4.1.4 API

In order for the API to be developed as quickly as possible, it needs to be as easy as possible to develop. In order for the API to be developed with as few changes as possible, it needs to be powerful enough to meet the needs of the system so that it can be adapted to as many situations as possible. But this raises the question of whether the API needs to be simple enough to be developed quickly or complex enough to be modified as little as possible. Efficient development and powerful functionality do not seem to go hand in hand.

Therefore, it is necessary to analyse the requirements in the design first. There are four basic database operations: add, check, modify, and delete; in the 3 design, there are only queries. So if the minimum number of APIs is one query API, but if you want one API to contain all query functions, the complexity of this interface may be greater than multiple interfaces. And when unique functionality is required for different functions that use this interface, it is possible that modifying the only API will affect the use of other functions, so the capabilities of the API can be split into different modules by function and scenario.

In order to satisfy the design in 3, there are three locations that need to be requested: (1) search for the starting node, which is not part of this study, but to obtain and display the node data in the database through the search box as required by the study, rather than adding a fixed node programmatically to the page code each time; (2) click on the node in the (3) click on a node in a non-modal dialogue box. The maximum number of APIs is, therefore, three. However, since the usage scenario for (1) is search, and the usage scenarios for (2) and (3) are to get the child node data by clicking on the node. Therefore, (2) and (3) can be combined into one API. Therefore, two APIs can be designed for the system, one for searching nodes and one for querying child nodes.

In summary, the specific functional descriptions and detailed descriptions of the APIs in the system are as follows:

Search Node API: This API is node data that can be obtained by node type and string for a node containing the corresponding keyword as a name. The form 4.1 is the request parameter for the search node API, and the form 4.2 is the return parameter for the search node API.

Parameters	**Required or not**	**Type**	**Description**
subject_type	Not	String	The type of node to be queried
sch_string	Not	String	String contained in the node name

Table 4.1: Request parameters for the Search Node API

Parameters	**Required or not**	**Type**	**Description**
subject_id	Yes	String	Node ID
subject_type	Yes	String	Types of nodes
subject_name	Yes	String	Name of the node

Table 4.2: Request parameters for the Search Node API

Child Node Query API: This API is an interface to get all the children of the corresponding node by their node ID. Forms4.3Query API request parameters for child nodes, Forms4.4Return parameters for the child node query API.

Parameters	Required or not	Type	Description
subject_id	Not	String	To query the node id of a child node

Table 4.3: Request parameters for the sub-node query API

Parameters	Required or not	Type	Description
predicate	Yes	String	Relationship to child nodes
object_id	Yes	String	Child Node ID
object_type	Yes	String	Type of child node
object_name	Yes	String	Name of the child node

Table 4.4: Return parameters for the sub-node query API

4.2 Front-end implementation

In this section, we look at front-end technologies, the modules that need to be implemented and the problems that need to be solved using front-end technologies. In the first section, the choice of front-end technology is explored, and the reasons for using React as the main technology for the front-end implementation. In 4.2.3 the different modules that need to be implemented and the issues that need to be addressed are explored.

4.2.1 Choice of front-end technology

Front-end content is the most direct point of interaction with the user, so the choice of front-end technology should not only consider the technical difficulty and technical feasibility, but also whether the visual interaction achieved by different technologies can meet the needs of this study.

The device used in this study is a normal home class laptop, so it can be used to display normal web information, but it does not have the ability to sacrifice a lot of performance for better results, so in the process of implementing the technology, it is necessary to minimize the system consumption. When updating large amounts of content using direct DOM node update techniques, it will consume more resources than React if no additional processing is done. React is currently a very popular JavaScript library, with over 120,000 Stars on Github and a very active presence on Stack Overflow. So there will be a wealth of material to work with. Therefore, the choice of React as the front-end development technology for this study is likely to be relatively good in terms of performance and potentially helpful in terms of knowledge and skill.

4.2.2 Component structure

React is a framework for JavaScript and requires a combination of these technologies to implement a component. There can be multiple components on a page, and components can be combined with each other.

New components with complex functionality can be formed by combining different components, the implementation of which is discussed in detail in 4.2.3. As the system to be implemented, the indented tree is both functionally and structurally tree-like, i.e. each node, whether root or child, is the same node, has the same structure and functionality, but presents different data, and can expand the whole result by a small node along a relationship, and the expanded new content can continue to expand according to the same rules. This is very similar to the recursive model 2.4. In 4.2, an indentation tree with this tree structure is implemented on the front end using a recursive algorithm. Using this tree as a base, other features of the knowledge graph can be combined in it, and because of its recursive nature, the whole tree can be continuously extended, with each newly extended node having the same extension capability.

4.2.3 Component implementation

In this section, we look at the components that make up the front end, the structure of the components and the technical issues that need to be addressed in their implementation. In the first section, indentation trees, recursion problems and implementations are discussed; in the second section, supernodes are discussed; in the third section, non-modal dialogue box implementations are discussed; in the fourth section, navigation implementations are discussed; and in the fifth section, search implementations are discussed.

Indented tree

The indented tree is the most important structure of the knowledge graph under study, and all other components need to be combined into this component. As discussed in 2.4 and 4.2.2, indented trees are recursive in nature. The nodes in the indented tree need to be scalable, so the indented tree expands outwards with one node as the smallest element, and each new node that expands is a scalable node.

As the indented tree may act as a root node or as a leaf node, some of the behaviour of this component is not fixed, for example, the root node does not have a relationship to the node before it, but other nodes do, and the node colour differs when different types of nodes are displayed. Therefore, a parameter configuration function has been added to this indented tree. When expanding a node, the backend will return node information, including the node type via the word node query API 4.4, which will be passed to the new indented tree which will then have a display that matches the node content.

infinite compilation: When trying to apply recursion to a node, I encountered a problem where the recursion would continue indefinitely until the program reported an error. The reason for this is that the recursive structure means that for an indented tree, all the operations it performs at load time will be performed by the indents nested within it, and if the indented tree nested within it is loaded at load time, then this loading behaviour will continue, so it is necessary to limit this loading behaviour. Ultimately, the infinite loading of the indentation tree is controlled by limiting the timing of recursive loading so that the default node displayed in the recursive tree is a non-recursive node, and the internal recursive tree is only loaded on the page when an event is triggered for the extended node.

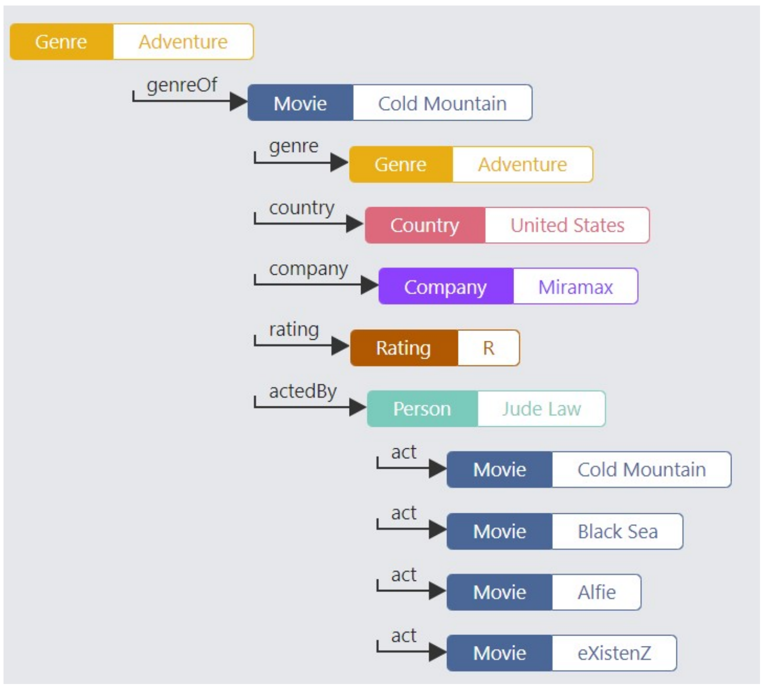


Figure 4.1: Implement: Tree

Super Nodes

Although a super node is a child node in an indented tree, it is not created when the number of nodes is small, so it is necessary to determine the child nodes when expanding the indented tree and then selectively use the super node component.

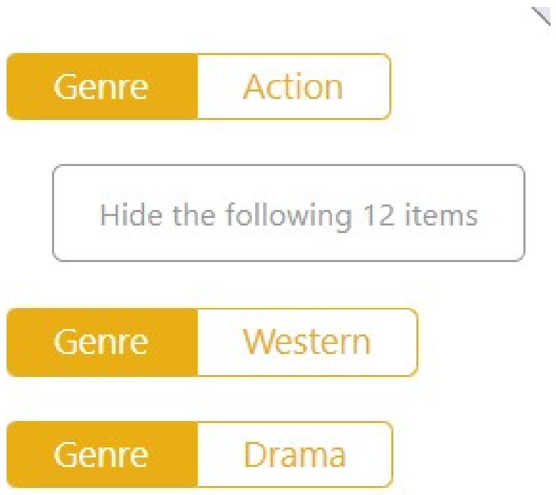


Figure 4.2: Implement of Super-Node

Non-modal dialogue

Non-modal dialogues are expanded by clicking on a node in the search result node list, and the position of the non-modal dialogue is not fixed but changes according to the position of the clicked points. By combining non-modal dialogues into nodes, each node has the ability to trigger a non-modal dialogue, but since nodes in non-modal dialogues do not need to pop up again, a configuration item is added to the non-modal dialogue component so that each node that expands in a non-modal dialogue is marked and identified so that it does not trigger a pop-up event for the non-modal dialogue, only the node extension event will be triggered.

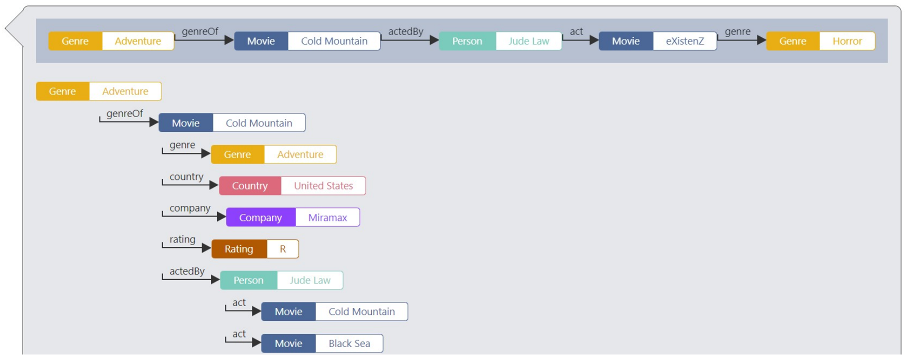


Figure 4.3: Implement: No-model Dialogue

Navigation

The navigation is displayed in a non-modal dialogue box to assist the user in obtaining the full path of the clicked node.

Get the full path: When trying to get the full path, we encountered the problem of not being able to use the same function to process the path. Since there is a parent-child relationship and a recursive relationship between the paths to be obtained, and to obtain the full path, one way to do this is to pass the node information up the hierarchy from the node that is triggered, with the parent passing its own pathfinding function to the child node as an argument each time it extends the child node. When a child node triggers a path-finding event, it executes the pathfinding function of the parent node and passes in the node information of the child node, accumulating online step by step. For this parent node, the algorithm executed when it is clicked should be: 4.2, and when its child nodes are clicked, the algorithm should be: 4.3. The code is very similar but different in that the same behaviour is performed by two separate functions, but the core difference is simply the number of parameters passed, which seems redundant. The implementation mechanism has been improved, as the nodes are tree structures and have to be expanded from the root node, so the path information is already generated during the expansion of the node and does not need to be computed on click. Therefore, a new 'path' property is added to each node, which records the full path from the starting node to the current node, and is retrieved from the current node's 'path' information when the path to a node is needed .

Listing 4.2: Code of a bad case of font-end

```
const myFunction = (myNode)=>{


    this.props.parentFunction(my);
};
```

Listing 4.3: Code of a bad case of font-end

```
const parentFunction = (childNode)=>{

    this.props.parentFunction(myNode + childNode);

};
```

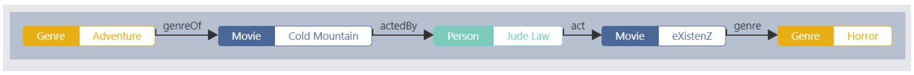


Figure 4.4: Implement: Navigation

4.3 Back-end and graph database

In this section, we will discuss the back-end and graph database implementation process. The data used in this system is the IMDB public data downloaded through Kaggle. In 4.3.1 discusses how to build the ontology, how to convert the data from tabular data to RDF format, and how to implement queries; in 4.3 explores how to use back-end technology Java to handle data requests. The focus of this research is on visual interaction, not on the construction of ontology, and not on solving problems with specific types of data, but on building ontology to support the task of visualising information, in order to find more general solutions for visualisation, i.e. to optimise interaction and visualisation, rather than to extract specific information from the data.

4.3.1 Data

This section discusses the construction of ontology, and uplift the data to RDF and Querying. In (73) a number of methods are used to construct ontology, including modelling them in 5 steps 2.1. However, as the focus of this study is on interaction and visualisation, rather than on studying specific data or finding specific answers through data, only some of these steps will be used. In Section 1, the implementation of Ontology modelling is discussed; in Section 2, the transformation of tabular data into RDF2.1 based on Ontology is discussed. In Section 3, we discuss how to store RDF data in a graph database and query the content from it.

Ontology modeling

As there is only one single page in this study, and the main information is in the indented tree, the system has fewer modules and fewer pages. Also, in order to further validate the interaction design in this study and to reduce the difficulty of the task due to the simplicity of the node relationships, rather than the user satisfaction with the interaction design, it is necessary to make the nodes and relationships in the knowledge graph as complex as possible, but not misleading, so that the approach discussed in 2.1 The approach to Ontology modelling explored in 2.1 needs to be modified.

In this study, Ontology modeling is accomplished in four steps:

Step 1: Identify the most appropriate field from the data to be used as an entity.

Step 2: Compare this entity with the currently selected entities in order to see if there is a one-way or two-way relationship.

Step 3: Return to step 1 until there are no more fields in the data table that can be selected.

Step 4: Draw a UML diagram.

Uplift

This section describes how to uplift '.CSV' table data to RDF2.1.

The model is 4.3.1 is first described in a '.ttl' file based on the model in 4.3.1, using the RDB to RDF Mapping Language (R2RML)(77), the data in this file is in RDF format, represented by a triple It is then processed through R2RML-F(15), and the program creates an RDF dataset based on the contents of R2RML and the data in the '.CSV' file. This dataset then stores the data needed for the graph database.

The R2RML code snippet is shown in 4.4,

Listing 4.4: Code snippet of R2RML

```
                ...
  <#Movie>
      a rr:TriplesMap ;
      rr:logicalTable [ rr:tableName "MOVIES" ] ;
      rr:subjectMap [
              rr:template "http://www.example.org/
```

```
                xiangmao/data/movie/{ID}" ;
                rr:class xmd:Movie;
        ];
                ...
```

Triplestore and Querying

Once an RDF dataset has been obtained via 4.3.1, it needs to be uploaded to a database that can store this triadic data. GraphDB is a graph database developed by Ontotext that can be used to store RDF triad datasets and query them via SPARQL(60). The code snippet for SPARQL is in 4.5.

Listing 4.5: SPARQL code snippets

```
                ...
    PREFIX xmd: <http://www.example.org/xiangmao/ont#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

    SELECT ?predicate ?object ?obj_type ?obj_name
    WHERE {
                ...
        ?object a ?obj_type;
        xmd:name ?obj_name.
    }
    ORDER BY(?object)
                ...
```

The figure shows the 4.5GraphDB database implementation.

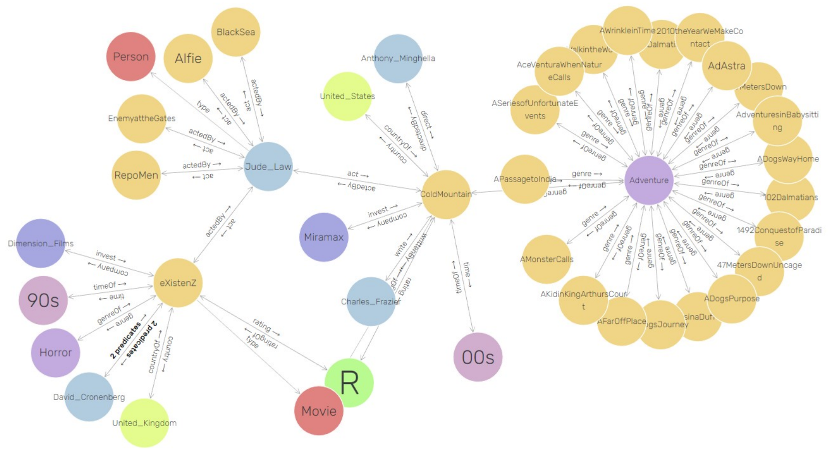


Figure 4.5: Implement: GraphDB

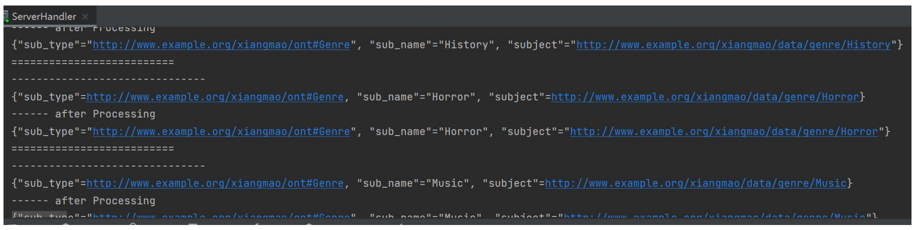
4.3.2 Processing data requests

This section describes how to implement responses to front-end data requests via Java, and requesting queries from the graph database.

When the front-end requests data from the back-end via API4.14.3, it processes the parameters passed to the back-end during the user request to determine the type of operation and the content to be queried. The parameters are combined with SPARQL query statements and parameters. The query results are then obtained through the Java library "Eclipse RDF4J"(76), which is a library for retrieving data from RDF databases.

The figure 4.6 shows the node information obtained from the back-end, and output in the console.

Figure 4.7 shows the overall system operation.



```
------ after Processing
{"sub_type"="http://www.example.org/xiangmao/ont#Genre", "sub_name"="History", "subject"="http://www.example.org/xiangmao/data/genre/History"}
==========================
------------------------------
{"sub_type"=http://www.example.org/xiangmao/ont#Genre, "sub_name"="Horror", "subject"=http://www.example.org/xiangmao/data/genre/Horror}
------ after Processing
{"sub_type"="http://www.example.org/xiangmao/ont#Genre", "sub_name"="Horror", "subject"="http://www.example.org/xiangmao/data/genre/Horror"}
==========================
------------------------------
{"sub_type"=http://www.example.org/xiangmao/ont#Genre, "sub_name"="Music", "subject"=http://www.example.org/xiangmao/data/genre/Music}
------ after Processing
{"sub_type"="http://www.example.org/xiangmao/ont#Genre", "sub_name"="Music", "subject"="http://www.example.org/xiangmao/data/genre/Music"}
```

Figure 4.6: Implement: Back-end

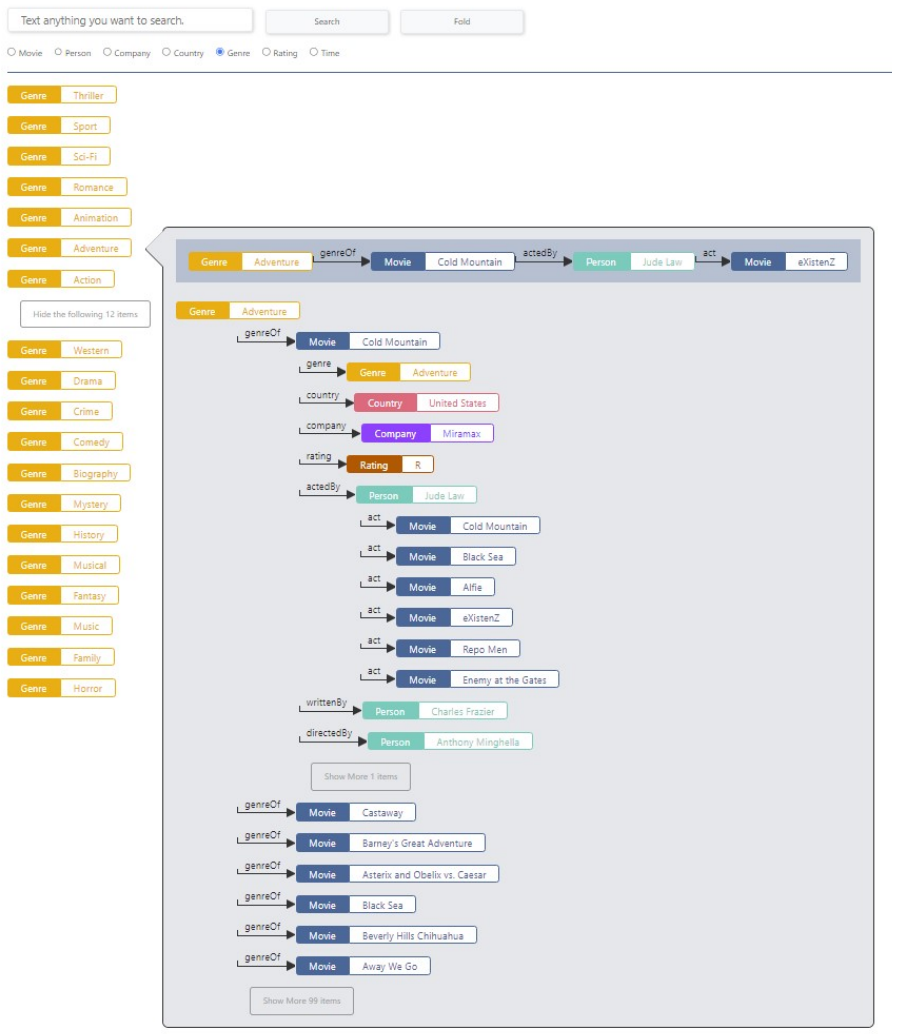


Figure 4.7: Implement: Overview

Chapter 5

Evaluation

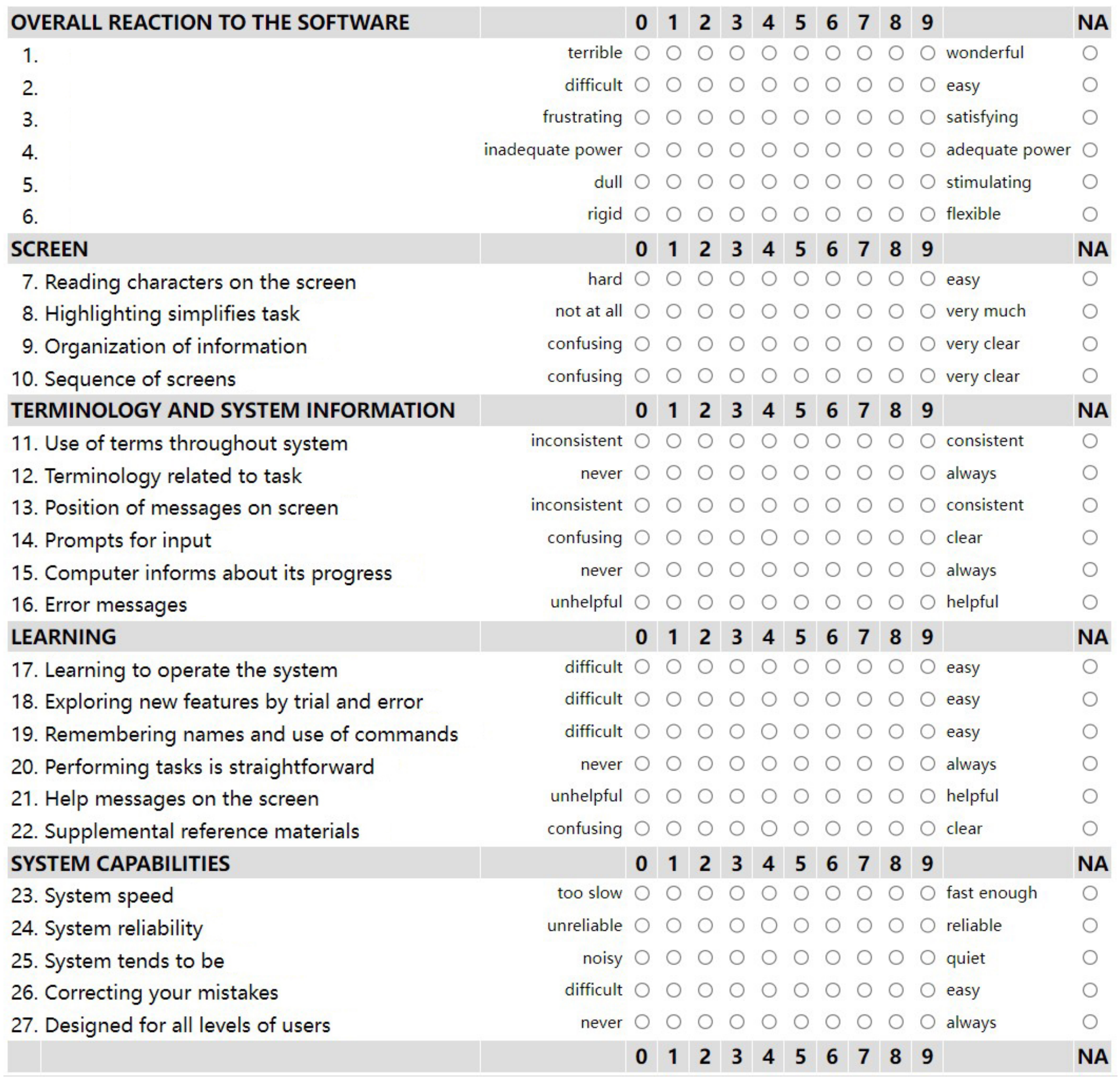
5.1 Background

5.1.1 Questionnaire for User Interface Satisfaction(QUIS)

QUIS stands for 'Questionnaire for User Interface Satisfaction, which is a questionnaire used to assess user satisfaction with an interface and can be used in research to understand or improve the efficiency of user interaction and user satisfaction with a system, for example: in the (34) For example, in the (34) study, because poor user interfaces can affect the use of computer systems, an attempt was made to design a system that would allow users to interact with the system more efficiently and with better satisfaction. The QUIS-short version, a standardized questionnaire for user interface satisfaction (12), was then used as one of the methods of user assessment.

Using the QUIS questionnaire, users are asked to complete five categories of 27 questions, which are rated on a scale of 10 from 0 to 9. The questionnaire was used to collect user satisfaction in the following five areas and to obtain a quantitative assessment of the user experience. Figure 5.1

1. OVERALL REACTIONS TO THE SOFTWARE
2. SCREEN
3. TERMINOLOGY AND SYSTEM INFORMATION
4. LEARNING
5. SYSTEM CAPABILITIES



OVERALL REACTION TO THE SOFTWARE		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
1.	terrible	○	○	○	○	○	○	○	○	○	○	wonderful	○
2.	difficult	○	○	○	○	○	○	○	○	○	○	easy	○
3.	frustrating	○	○	○	○	○	○	○	○	○	○	satisfying	○
4.	inadequate power	○	○	○	○	○	○	○	○	○	○	adequate power	○
5.	dull	○	○	○	○	○	○	○	○	○	○	stimulating	○
6.	rigid	○	○	○	○	○	○	○	○	○	○	flexible	○
SCREEN		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
7. Reading characters on the screen	hard	○	○	○	○	○	○	○	○	○	○	easy	○
8. Highlighting simplifies task	not at all	○	○	○	○	○	○	○	○	○	○	very much	○
9. Organization of information	confusing	○	○	○	○	○	○	○	○	○	○	very clear	○
10. Sequence of screens	confusing	○	○	○	○	○	○	○	○	○	○	very clear	○
TERMINOLOGY AND SYSTEM INFORMATION		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
11. Use of terms throughout system	inconsistent	○	○	○	○	○	○	○	○	○	○	consistent	○
12. Terminology related to task	never	○	○	○	○	○	○	○	○	○	○	always	○
13. Position of messages on screen	inconsistent	○	○	○	○	○	○	○	○	○	○	consistent	○
14. Prompts for input	confusing	○	○	○	○	○	○	○	○	○	○	clear	○
15. Computer informs about its progress	never	○	○	○	○	○	○	○	○	○	○	always	○
16. Error messages	unhelpful	○	○	○	○	○	○	○	○	○	○	helpful	○
LEARNING		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
17. Learning to operate the system	difficult	○	○	○	○	○	○	○	○	○	○	easy	○
18. Exploring new features by trial and error	difficult	○	○	○	○	○	○	○	○	○	○	easy	○
19. Remembering names and use of commands	difficult	○	○	○	○	○	○	○	○	○	○	easy	○
20. Performing tasks is straightforward	never	○	○	○	○	○	○	○	○	○	○	always	○
21. Help messages on the screen	unhelpful	○	○	○	○	○	○	○	○	○	○	helpful	○
22. Supplemental reference materials	confusing	○	○	○	○	○	○	○	○	○	○	clear	○
SYSTEM CAPABILITIES		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
23. System speed	too slow	○	○	○	○	○	○	○	○	○	○	fast enough	○
24. System reliability	unreliable	○	○	○	○	○	○	○	○	○	○	reliable	○
25. System tends to be	noisy	○	○	○	○	○	○	○	○	○	○	quiet	○
26. Correcting your mistakes	difficult	○	○	○	○	○	○	○	○	○	○	easy	○
27. Designed for all levels of users	never	○	○	○	○	○	○	○	○	○	○	always	○
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**

Figure 5.1: Questionnaire for User Interface Satisfaction(QUIS)

5.1.2 Ethical review

The process and methods of data collection in this trial have been subject to a Research Ethics Review by Trinity College Dublin. The details of the Research Ethics Review are discussed in detail in section 5.1.3.

5.1.3 Research Ethics Review

How to recruit

By random selection in the flat lobby.

Number of participants

Greater than or equal to 20 people. This Survey will use QUIS(Questionnaire for User Interface Satisfaction). The minimum sample size for a QUIS survey is 20.

Type of participant

Those who are able to complete the questionnaire using a PC properly.

Procedures of the study Participants will be asked to use a knowledge graph application on a PC and to complete a survey based on their own experience of using it.

Debriefing arrangements following the study

At the end of the experiment, I will explain more of the reasons behind the experiment and give them a chance to ask questions.

Describe any potential benefits and potential harms to participants, and how they are addressed.

I do not anticipate any risks to participants. While participation will not benefit the participants directly, the research will support the development of better tools for Knowledge Graph, which will benefit Users of Knowledge Graph.

Describe any conflicts of interest and how they will be addressed.

- It is possible that some of the participants had some personal relationship with the researcher. The results of the overall questionnaire will be used to measure the effectiveness of the methodology used in the researcher's Master's Dissertation.
- Conflicts of interest are resolved by actively recusing users who have a personal relationship with the researcher from participating in the survey.

5.2 Evaluation Methodology

As described in 5.1.1, this study will use QUIS as a research tool to collect user satisfaction and use it to evaluate the effectiveness of the interaction optimisation in this Dissertation.

5.2.1 Functional scope

In order to cover the interaction optimisation in this Dissertation, users were assigned tasks that were selected to display the optimisation items covered in this Dissertation, as some interactions were only displayed on the page under certain circumstances.

Some of the features that can only be evaluated in certain circumstances are:

indentation tree: The indentation tree is evaluated when the user has completed the search for a node with a level greater than or equal to 2. If the user needs to find a node at the first level, the target node will be found, and the task will be finished before the user expands the node using the indentation tree, in which case there is no need for the user to use the indentation tree.

Non-modal dialogue : Non-modal dialogues need to be evaluated before the user can finish searching for nodes with a level greater than or equal to 2.

Navigation: Navigation is required when the node the user is looking for is in a modal dialogue.

SuperNode: SuperNode requires the user to expand a node with more than seven children while searching for the target node.

In this study, to cover all of the above optimisations, the task is chosen to satisfy all of the above conditions, and the task is to find a node with a depth of 6 levels and more than 100 children in its path.

5.2.2 Task

The task is to compare the same movie genres found by the following two paths through the tree knowledge graph, where the names of the relationships between the nodes are enclosed in square brackets ”[]”.

- Path I: ”Genre:Adventure” -> [”GenreOf”] -> ”Movie:Cold Mountain” -> [”actedBy”] -> ”Person:Jude Law” -> [”act”] -> ”Movie: eXistenZ” -> [”genre”] -> ”Genre:Horror”.
- Path 2: ”Genre:History” -> [”genreOf”] -> ”Movie:The Bussiness of Show Business” -> [”country”] -> ”Country:Canada” -> [”countryOf”] -> ”Movie:Prom Night” -> [”genre”] -> ”Genre:horror”.

5.2.3 Participation process

Volunteers were recruited in the lobby of Student Accommodation, where they were asked about the experiment and informed of their rights and how to participate. The average age of the participants was between 20 and 30 years old, and they all had basic computer knowledge and experience in using a web browser. Volunteers will be assigned to one of two sets of tasks as users of the system, and users who have participated in one set of tasks will not be able to participate in the other. The whole process of user participation is usually completed in less than 5 minutes, and no user has used the system for more than 10 minutes.

5.3 Results

The total sample size was 25, and all participants were between the ages of 20 and 30.

The table 5.1 is the result of QUIS, which consists of 5 sections, assessing ”OVERALL REACTION TO THE SOFTWARE”, ”SCREEN”, ”TERMINOLOGY AND SYSTEM INFORMATION”, ”LEARNING”, and ”SYSTEM CAPABILITIES”.

Where AVG is the average score for each question, and the AVG of the TOTAL item is the sum of the average scores for each question in the section; where MID is the median score for each question, and the MID of the TOTAL item is the sum of the median scores for each question in the section; where MAX is the maximum score for each question, and the MAX of the TOTAL item is the sum of the maximum scores for each question in the section; where MINI is the lowest score for each question and MINI for TOTAL is the sum of the lowest scores for each question in the section;

Figure 5.2 for, Table5.1 for the comparison of the total scores of 4 items out of 5 sections.

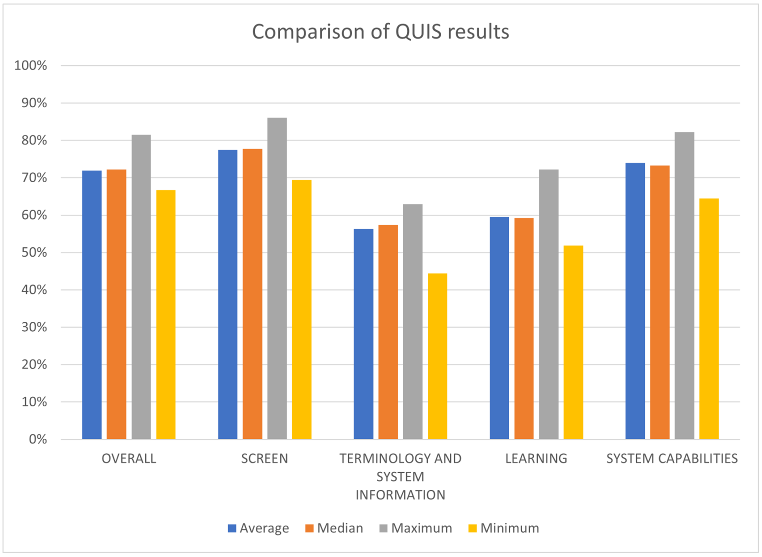


Figure 5.2: The comparison of the total scores of 4 items out of 5 sections.

5.4 Discussion

5.4.1 Overall Reaction to the software

The overall mean score was similar to the overall median score, with both scores exceeding 70%. The highest mean score among all questions was ”dull OR stimulating” with a score of 6.76; the lowest mean score was ”terrible OR wonderful” with a score of 5.76.

Users rated ”dull OR stimulating” more positively than other factors in the overall rating, but rated ”terrible OR wonderful” more negatively, so they seem to be more inclined to think that the interface is fun, but not very user-friendly.

5.4.2 Screen

This section was used to assess satisfaction with the readability of the information displayed on the screen. The overall mean score was similar to the overall median score, 77% and 78% respectively, with the highest mean score for all questions being ”Reading characters on the screen.) ”, with a score of 6.48.

Users seem to be most satisfied with the recognition of characters on the screen compared to other aspects of the screen display, and the Sequence of screens can be confusing for users.

5.4.3 Terminology and System information

This section was used to assess whether the terminology and system information used on the screen was satisfactory to the user. The overall mean and median scores were similar, at 56% and 57% respectively, with the highest mean score of all questions being ”Use of terms throughout the system.” at 6.60 and the lowest mean score being ”Error messages.

The overall score for this section is low, with ”Error messages.” scoring the lowest and ”Computer informs about its progress.” also scoring low at 2.88. Even though users can indirectly see the progress of a task and whether an error has been made by viewing the navigation, they seem to be dissatisfied with the lack of valuable and more direct error messages after an error has been made, and seem to be dissatisfied with the usefulness of the navigation as an aid to ”Computer informs about its progress.

5.4.4 Learning

This section assesses the satisfaction of users in learning the rules of the system on their own after they have been introduced to it. The highest average score was 7.28 for ”Exploring new features by trial and error.” Users also scored relatively high in this section with a score of 6.96 for the other operation-related item, ”Learning to operate the system. The section ”Help messages and supplementary material to assist users in learning the system. Help messages on the screen.” and ”Supplemental reference materials.” with scores of 2.28 and 1.84, respectively.

In terms of learning, the system seems to have some effect in terms of learning the rules of the system by using the trial function, but the information provided to the user for learning and understanding the system does not seem to satisfy the user.

5.4.5 System Capabilities

This section assesses the capabilities of the system, which include two main areas of capability: hardware capability and design capability. For the system, ”System tends to be. (noisy - quiet)” scored 6.96 and ”System speed. (too slow - fast enough)” scored 6.32. Users seem to be dissatisfied with the speed of the system. The average score for ”Designed for all levels of users.” is 6.36, with a maximum score of 8. It seems that some users think that the system is designed for all types of users.

OVERALL REACTION TO THE SOFTWARE					
		AVG	**MID**	**MAX**	**MINI**
	TOTAL	38.84 (72%)	39.00 (72%)	44.00 (81%)	36.00 (67%)
1	terrible OR wonderful	5.76	6.00	8.00	4.00
2	difficult OR easy	6.44	6.00	9.00	5.00
3	frustrating OR satisfying	6.56	7.00	8.00	6.00
4	inadequate power OR adequate power	6.72	7.00	8.00	6.00
5	dull OR stimulating	6.76	7.00	8.00	6.00
6	rigid OR flexible	6.60	7.00	9.00	6.00
SCREEN					
		AVG	**MID**	**MAX**	**MINI**
	TOTAL	27.88 (77%)	28.00 (78%)	31.00 (86%)	25.00 (69%)
7	Reading characters on the screen.	7.48	7.00	8.00	7.00
8	Highlighting simplifies task.	6.80	6.00	8.00	6.00
9	Organization of information.	7.12	8.00	8.00	6.00
10	Sequence of screens.	6.48	6.00	7.00	6.00
TERMINOLOGY AND SYSTEM INFORMATION					
		AVG	**MID**	**MAX**	**MINI**
	TOTAL	30.44 56%	31.00 57%	34.00 63%	24.00 44%
11	Use of terms throughout system.	6.60	7.00	7.00	6.00
12	Terminology related to task.	6.28	6.00	7.00	4.00
13	Position of messages on screen.	6.08	6.00	7.00	4.00
14	Prompts for input.	6.20	7.00	7.00	4.00
15	Computer informs about its progress.	2.88	3.00	5.00	2.00
16	Error messages.	2.40	2.00	3.00	1.00
LEARNING					
		AVG	**MID**	**MAX**	**MINI**
	TOTAL	32.12 (59%)	32.00 (59%)	39.00 (72%)	28.00 (52%)
17	Learning to operate the system.	6.96	7.00	8.00	6.00
18	Exploring new features by trial and error.	7.28	7.00	8.00	6.00
19	Remembering names and use of commands.	6.92	7.00	9.00	6.00
20	Performing tasks is straightforward.	6.84	7.00	9.00	4.00
21	Help messages on the screen.	1.84	2.00	5.00	0.00
22	Supplemental reference materials.	2.28	2.00	6.00	0.00
SYSTEM CAPABILITIES					
		AVG	**MID**	**MAX**	**MINI**
	TOTAL	33.28 (74%)	33.00 (73%)	37.00 (82%)	29.00 (64%)
23	System speed.	6.32	7.00	8.00	4.00
24	System reliability.	6.92	7.00	8.00	5.00
25	System tends to be.	6.96	7.00	8.00	6.00
26	Correcting your mistakes.	6.72	7.00	8.00	6.00
27	Designed for all levels of users.	6.36	6.00	8.00	0.00

Table 5.1: The comparison of the total scores of 4 items out of 5 sections.

Chapter 6

Conclusions & Future Work

6.1 Conclusions

In this study, we reviewed the literature on visual interactions, knowledge graphs and related implementation techniques, selected and optimised the interactions, applied them to the design, selected the techniques used to implement the system, implemented an indented tree with a recursive structure through a front-end technique, solved the encountered problem, and finally evaluated it through QUIS. It was found that the knowledge graph seemed to perform better in terms of overall satisfaction than other aspects of satisfaction in QUIS and that even when the overall performance was less than fantastic, it was still not boring to the user. Although the Knowledge Graph does not perform better than other aspects of the screen display in terms of Sequence of screens, the user has no difficulty in reading the information on the screen. Although the system's error messages may not be satisfactory to all users, there is still consistent terminology throughout the system, e.g. node types and names of relationships. And this level of uniformity in terminology does not seem to be worse than error messages in the current sample. Users seem to think that it is not easy to learn the system from help and reference material, but experimenting with this approach to understanding the system seems to be effective. Thus, the results seem to suggest that the advantages of using the Knowledge Graph to optimise interactions, even if not perfect, can compensate to some extent for the disadvantages and make the interactions still effective.

6.2 Future Work

In this Dissertaion, although some optimisations have been made in the area of interaction, two problems remain:

Scenario, not typical: The data in this Dissertation is IMDB movie data, movie-related information as some movies are familiar to many people, so although attempts have been made to make the Ontology and predicate relationships created from this dataset as complex as possible by modifying the Ontology modelling step, and selecting cold information for the evaluation task, in order to reduce the influence of the user's familiarity with the dataset on the evaluation results. However, it is still not possible to exclude the influence completely.

lack of group control: In this experiment, only the completed knowledge graphs were evaluated, but it is not particularly clear how the different optimisation methods affect the evaluation of user satisfaction, and there may be other confounding factors.

Therefore, future experiments could be conducted using sufficiently typical or cold datasets or finding an Ontology modelling approach that can mask the effect of different datasets on the results. More grouped controlled experiments could also be added, where only one optimisation change is implemented for each control group at a time, to find the impact of different interactive elements on user experience and user satisfaction by comparing multiple groups.

Bibliography

[1] Alani, H. (2003). Tgviztab: An ontology visualisation extension for protégé.

[2] Allan, J., Croft, B., Moffat, A., and Sanderson, M. (2012). Frontiers, challenges, and opportunities for information retrieval: Report from swirl 2012 the second strategic workshop on information retrieval in lorne. In *Acm sigir forum*, volume 46, pages 2–32. ACM New York, NY, USA.

[3] Arenas, M., Grau, B. C., Kharlamov, E., Marciuška, Š., and Zheleznyakov, D. (2016). Faceted search over rdf-based knowledge graphs. *Journal of Web Semantics*, 37:55–74.

[4] Auer, S., Kovtun, V., Prinz, M., Kasprzik, A., Stocker, M., and Vidal, M. E. (2018). Towards a knowledge graph for science. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, pages 1–6.

[5] Bibeault, B., De Rosa, A., and Katz, Y. (2015). *jQuery in Action*. Simon and Schuster.

[6] Bierman, G., Abadi, M., and Torgersen, M. (2014). Understanding typescript. In *European Conference on Object-Oriented Programming*, pages 257–281. Springer.

[7] Butler, M. and Robila, S. (2016). Interface for querying and data mining for the imdb dataset. In *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–6. IEEE.

[8] Cao, N., Lin, Y.-R., Sun, X., Lazer, D., Liu, S., and Qu, H. (2012). Whisper: Tracing the spatiotemporal process of information diffusion in real time. *IEEE transactions on visualization and computer graphics*, 18(12):2649–2658. ISBN: 1077-2626 Publisher: IEEE.

[9] Card, M. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann.

[10] Carlson, C. N. (2003). Information overload, retrieval strategies and internet user empowerment.

[11] Carter, L. F. (1947). Presenting Numerical Data by the Use of Tables and Graphs'. *Psychological research on equipment design*, (19):65. ISBN: 0598914838 Publisher: US Government Printing Office.

[12] Chin, J. P., Diehl, V. A., and Norman, K. L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–218.

[13] Cui, W., Liu, S., Tan, L., Shi, C., Song, Y., Gao, Z., Qu, H., and Tong, X. (2011). Textflow: Towards better understanding of evolving topics in text. *IEEE transactions on visualization and computer graphics*, 17(12):2412–2421. ISBN: 1077-2626 Publisher: IEEE.

[14] Culbertson, H. M. and Powers, R. D. (1959). A study of graph comprehension difficulties. *Audio Visual Communication Review*, pages 97–110. ISBN: 0885-727X Publisher: JSTOR.

[15] Debruyne, C. and O'Sullivan, D. (2016). R2rml-f: towards sharing and executing domain logic in r2rml mappings. In *LDOW@ WWW*.

[16] Decker, S., Mitra, P., and Melnik, S. (2000). Framework for the semantic web: an rdf tutorial. *IEEE Internet Computing*, 4(6):68–73.

[17] Del R. Millán, J., Ferrez, P. W., Galán, F., Lew, E., and Chavarriaga, R. (2008). NON-INVASIVE BRAIN-MACHINE INTERACTION. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(05):959–972.

[18] Dictionaries, O. (2015). Oxford dictionaries. *Oxford University Press) Retrieved March*, 7:2015.

[19] Dörk, M., Riche, N. H., Ramos, G., and Dumais, S. (2012). Pivotpaths: Strolling through faceted information spaces. *IEEE transactions on visualization and computer graphics*, 18(12):2709–2718. ISBN: 1077-2626 Publisher: IEEE.

[20] Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2.

[21] Eick, S. G. and Karr, A. F. (2002). Visual scalability. *Journal of Computational and Graphical Statistics*, 11(1):22–43. ISBN: 1061-8600 Publisher: Taylor & Francis.

[22] Epp, S. S. (2010). *Discrete mathematics with applications.* Cengage learning.

[23] Färber, M., Bartscherer, F., Menne, C., and Rettinger, A. (2018). Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, 9(1):77–129.

[24] Feather, J. and Sturges, P. (2003). *International encyclopedia of information and library science*. Routledge.

[25] Fox, T. A., Heimendinger, J., and Block, G. (1992). Telephone surveys as a method for obtaining dietary information: a review. *Journal of the American Dietetic Association*, 92(6):729–733.

[26] Fu, B., Noy, N. F., and Storey, M.-A. (2013). Indented tree or graph? a usability study of ontology visualization techniques in the context of class mapping evaluation. In *International Semantic Web Conference*, pages 117–134. Springer.

[27] Graham, R. L., Knuth, D. E., Patashnik, O., and Liu, S. (1989). Concrete mathematics: a foundation for computer science. *Computers in Physics*, 3(5):106–107.

[28] Gupta, S., Kaiser, G., Neistadt, D., and Grimm, P. (2003). Dom-based content extraction of html documents. In *Proceedings of the 12th international conference on World Wide Web*, pages 207–214.

[29] Head, A. and Eisenberg, M. (2011). How college students use the web to conduct everyday life research. *First Monday*, 16(4).

[30] Healey, C. G. and Dennis, B. M. (2012). Interest driven navigation in visualization. *IEEE transactions on visualization and computer graphics*, 18(10):1744–1756. ISBN: 1077-2626 Publisher: IEEE.

[31] Herman, I., Melançon, G., and Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6(1):24–43.

[32] Hölscher, C. and Strube, G. (2000). Web search behavior of internet experts and newbies. *Computer networks*, 33(1-6):337–346.

[33] Huang, W., Eades, P., and Hong, S.-H. (2009). Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization*, 8(3):139–152.

[34] Jaspers, M. W., Steen, T., Van Den Bos, C., and Geenen, M. (2004). The think aloud method: a guide to user interface design. *International journal of medical informatics*, 73(11-12):781–795.

[35] Jensen, S. H., Møller, A., and Thiemann, P. (2009). Type analysis for javascript. In *International Static Analysis Symposium*, pages 238–255. Springer.

[36] Jotheeswaran, J. and Kumaraswamy, D. Y. S. (2005). OPINION MINING USING DECISION TREE BASED FEATURE SELECTION THROUGH MANHATTAN HIERARCHICAL CLUSTER MEASURE. . *Vol.*, 58:9.

[37] Keim, D. A., Mansmann, F., Schneidewind, J., and Ziegler, H. (2006). Challenges in visual data analysis. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 9–16. IEEE.

[38] Kontiza, K., Bikakis, A., and Miller, R. (2015). Cognitive-based visualization of semantically structured cultural heritage data. In *CEUR Workshop Proceedings*, volume 1456, pages 61–68. CEUR.

[39] Krahenbuhl, J. H. (2015). *Learning Axure RP interactive prototypes.* Packt Publishing Ltd.

[40] Kuhail, M. A., Negreiros, J., and Seffah, A. (2021). Teaching recursive thinking using unplugged activities. *World Transactions on Engineering and Technology Education*, 19(2):169.

[41] Kulahcioglu, T., Fradkin, D., Parlak, A., and Belkov, A. (2020). Logvis: Graph-assisted visual analysis of event logs from industrial equipment. In *VOILA@ ISWC*, pages 61–72.

[42] Kurata, Y. and Egenhofer, M. J. (2005). Semantics of simple arrow diagrams. In *AAAI Spring Symposium: Reasoning with Mental and External Diagrams: Computational Modeling and Spatial Assistance*, pages 101–104.

[43] Lam, H. (2008). A framework of interaction costs in information visualization. *IEEE transactions on visualization and computer graphics*, 14(6):1149–1156. ISBN: 1077-2626 Publisher: IEEE.

[44] Langford, L. K. (2010). Surf's up: harnessing information overload. *IEEE Engineering Management Review*, 38(1):164–165.

[Larkin and Simon] Larkin, J. H. and Simon, H. Why a Diagram is (Sometimes)Worth Ten ThousandWords. page 36.

[46] Lewandowski, D. (2008). The retrieval effectiveness of web search engines: considering results descriptions. *Journal of documentation.*

[47] Li, Z., Zhao, S., Duan, J., Su, C.-Y., Yang, C., and Zhao, X. (2017). Human Cooperative Wheelchair With Brain–Machine Interaction Based on Shared Control Strategy. *IEEE/ASME Transactions on Mechatronics*, 22(1):185–195.

[48] Liu, S., Zhou, M. X., Pan, S., Qian, W., Cai, W., and Lian, X. (2009). Interactive, topic-based visual text summarization and analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 543–552.

[49] Liu, W., Cao, Y., and Proctor, R. W. (2021). How do app icon color and border shape influence visual search efficiency and user experience? evidence from an eye-tracking study. *International Journal of Industrial Ergonomics*, 84:103160.

[50] Lohse, J. (1991). A cognitive model for the perception and understanding of graphs. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 137–144.

[51] Ma, J., Liao, I., Ma, K.-L., and Frazier, J. (2012). Living liquid: Design and evaluation of an exploratory visualization tool for museum visitors. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2799–2808. ISBN: 1077-2626 Publisher: IEEE.

[52] Maslow, A. H. (1981). *Motivation and personality.* Prabhat Prakashan.

[Melville et al.] Melville, P., Mooney, R. J., and Nagarajan, R. Content-Boosted Collaborative Filtering for Improved Recommendations. page 6.

[54] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81.

[55] Munzner, T. (2009). A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6):921–928. ISBN: 1077-2626 Publisher: IEEE.

[56] Munzner, T., Guimbretiere, F., Tasiran, S., Zhang, L., and Zhou, Y. (2003). Treejuxtaposer: scalable tree comparison using focus+ context with guaranteed visibility. In *ACM SIGGRAPH 2003 Papers*, pages 453–462.

[57] Ofoeda, J., Boateng, R., and Effah, J. (2019). Application programming interface (api) research: A review of the past to inform the future. *International Journal of Enterprise Information Systems (IJEIS)*, 15(3):76–95.

[58] Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.

[59] Peissner, M., Doebler, V., and Metze, F. (2011). Can voice interaction help reducing the level of distraction and prevent accidents. *Meta-Study Driver Distraction Voice Interaction*, 24(5).

[60] Pérez, J., Arenas, M., and Gutierrez, C. (2009). Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45.

[61] Pinker, S. (1990). A theory of graph comprehension. *Artificial intelligence and the future of testing*, 73:126.

[62] Raggett, D., Le Hors, A., Jacobs, I., et al. (1999). Html 4.01 specification. *W3C recommendation*, 24.

[63] Riener, A., Ferscha, A., Bachmair, F., Hagmüller, P., Lemme, A., Muttenthaler, D., Pühringer, D., Rogner, H., Tappe, A., and Weger, F. (2013). Standardization of the in-car gesture interaction space. In *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 14–21.

[64] Rubin, R. E. (2017). *Foundations of library and information science.* American Library Association.

[65] Sarrafzadeh, B., Vechtomova, O., and Jokic, V. (2014). Exploring knowledge graphs for exploratory search. In *Proceedings of the 5th Information Interaction in Context Symposium*, pages 135–144.

[66] Sarrafzadeh, B., Vtyurina, A., Lank, E., and Vechtomova, O. (2016). Knowledge graphs versus hierarchies: An analysis of user behaviours and perspectives in information seeking. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, pages 91–100.

[67] Saunders, P. C. and Interrante, V. (2006). An investigation into color preference for color palette selection in multivariate visualization. *Univ. Minneapolis, DTC (Digital Technology Center) Research Report*, 15:2006.

[68] Schermer, M. (2009). The Mind and the Machine. On the Conceptual and Moral Implications of Brain-Machine Interaction. *NanoEthics*, 3(3):217–230.

[Schickel and Faltings] Schickel, V. and Faltings, B. Using an Ontologcial A-priori Score to Infer User' s Preferences. page 5.

[70] Schober, M. F. (2018). The future of face-to-face interviewing. *Quality Assurance in Education.*

[71] Schutz, H. G. (1961). An evaluation of formats for graphic trend displays—experiment II. *Human Factors*, 3(2):99–107. ISBN: 0018-7208 Publisher: SAGE Publications Sage CA: Los Angeles, CA.

[72] Scott, G. P., Shah, P., Wyatt, J. C., Makubate, B., and Cross, F. W. (2011). Making electronic prescribing alerts more effective: scenario-based experimental study in junior doctors. *Journal of the American Medical Informatics Association*, 18(6):789–798.

[73] Shimizu, C., Hitzler, P., Hirt, Q., Rehberger, D., Estrecha, S. G., Foley, C., Sheill, A. M., Hawthorne, W., Mixter, J., Watrall, E., et al. (2020). The enslaved ontology: Peoples of the historic slave trade. *Journal of Web Semantics*, 63:100567.

[74] Shiravi, H., Shiravi, A., and Ghorbani, A. A. (2011). A survey of visualization systems for network security. *IEEE Transactions on visualization and computer graphics*, 18(8):1313–1329.

[75] Sinha, G., Shahi, R., and Shankar, M. (2010). Human computer interaction. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 1–4. IEEE.

[76] Sotona, A., Negru, S., Prague, M., et al. (2016). How to feed apache hbase with petabytes of rdf data: An extremely scalable rdf store based on eclipse rdf4j. In *Proceedings of the ISWC.*

[77] Souripriya Das, Seema Sundara, R. C. (2012). R2RML: RDB to RDF Mapping Language.

[78] Sturges, J. E. and Hanrahan, K. J. (2004). Comparing telephone and face-to-face qualitative interviewing: a research note. *Qualitative research*, 4(1):107–118.

[79] Suglia, A., Greco, C., Musto, C., de Gemmis, M., Lops, P., and Semeraro, G. (2017). A Deep Architecture for Content-based Recommendations Exploiting Recurrent Neural Networks. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 202–211, Bratislava Slovakia. ACM.

[80] Tidwell, J. (2010). *Designing interfaces: Patterns for effective interaction design.* " O'Reilly Media, Inc.".

[81] Tietz, T., Jäger, J., Waitelonis, J., and Sack, H. (2016). Semantic annotation and information visualization for blogposts with refer. In *VOILA@ ISWC*, pages 28–40.

[82] Tominski, C., Forsell, C., and Johansson, J. (2012). Interaction support for visual comparison inspired by natural behavior. *IEEE Transactions on visualization and computer graphics*, 18(12):2719–2728. ISBN: 1077-2626 Publisher: IEEE.

[83] Tu, Y. and Shen, H.-W. (2008). Balloon focus: a seamless multi-focus+ context method for treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1157–1164. ISBN: 1077-2626 Publisher: IEEE.

[84] Tufte, E. R., Goeler, N. H., and Benson, R. (1990). *Envisioning information*, volume 126. Graphics press Cheshire, CT.

[85] Tversky, B., Zacks, J., Lee, P., and Heiser, J. (2000). Lines, blobs, crosses and arrows: Diagrammatic communication with schematic figures. In *International conference on theory and application of diagrams*, pages 221–230. Springer.

[86] Washburne, J. N. (1927). An experimental study of various graphic, tabular, and textual methods of presenting quantitative material. *Journal of Educational Psychology*, 18(6):361.

[87] White, R. W. and Roth, R. A. (2009). Exploratory search: Beyond the query-response paradigm. *Synthesis lectures on information concepts, retrieval, and services*, 1(1):1–98.

[88] Wong, B. (2011). Color blindness. *nature methods*, 8(6):441.

[89] Wu, Y., Wei, F., Liu, S., Au, N., Cui, W., Zhou, H., and Qu, H. (2010). OpinionSeer: interactive visualization of hotel customer feedback. *IEEE transactions on visualization and computer graphics*, 16(6):1109–1118. ISBN: 1077-2626 Publisher: IEEE.

[90] Yi, J. S., ah Kang, Y., Stasko, J., and Jacko, J. A. (2007). Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–1231. ISBN: 1077-2626 Publisher: IEEE.

[91] Yoghourdjian, V., Archambault, D., Diehl, S., Dwyer, T., Klein, K., Purchase, H. C., and Wu, H.-Y. (2018). Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics*, 2(4):264–282.

[92] Zhao, X., Chen, H., Xing, Z., and Miao, C. (2021). Brain-inspired search engine assistant based on knowledge graph. *IEEE Transactions on Neural Networks and Learning Systems*.

Appendix

GitHub: github.com/iczcpkqo/Dissertation

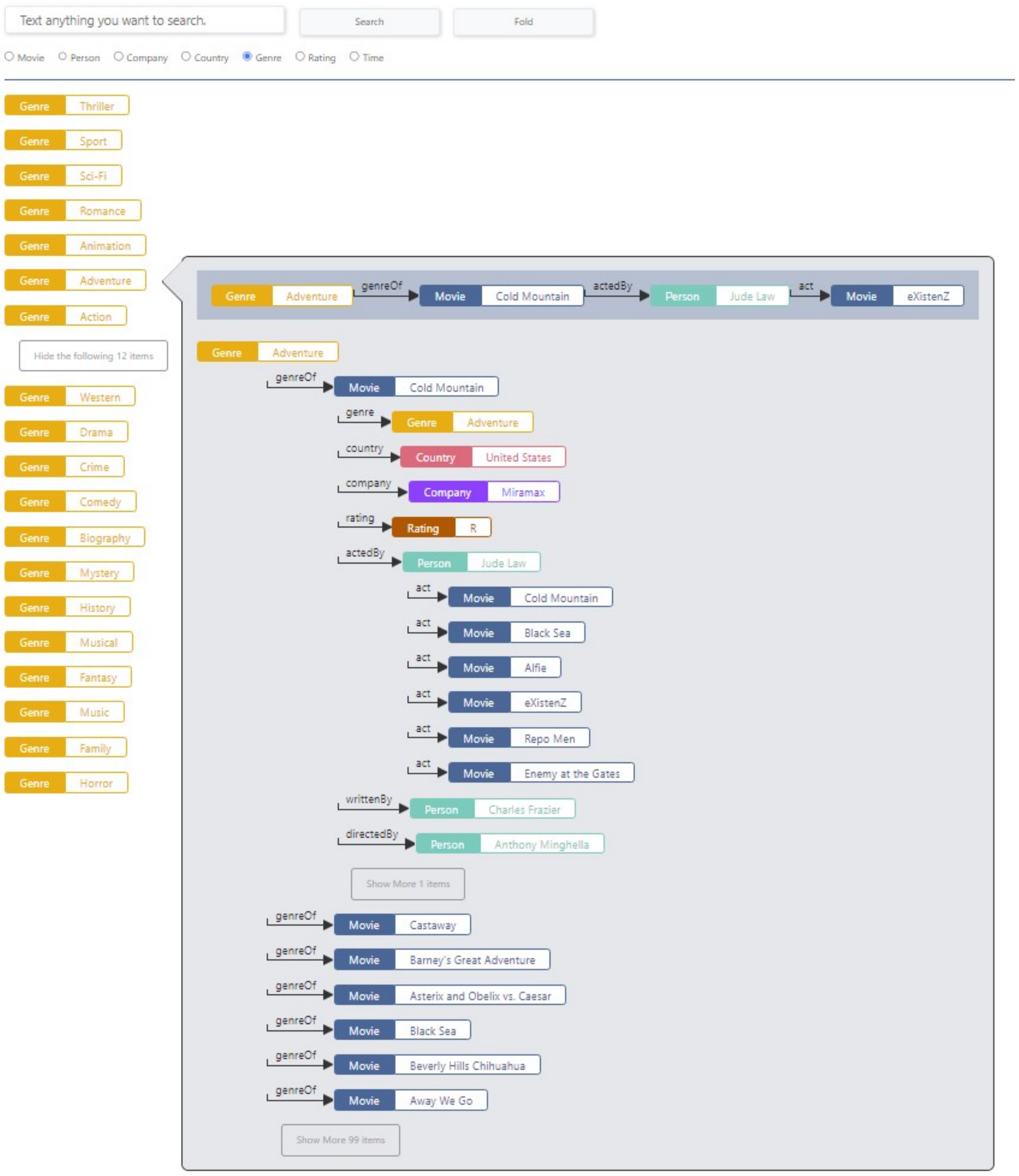


Figure 1: Overview