

Visualization Assisted Enterprise Search Engine

Saiful Khan

New College



Michaelmas Term 2015

A thesis submitted for the degree of
Doctor of Philosophy



Department of Engineering Science
University of Oxford

Visualization Assisted Enterprise Search Engine

Saiful Khan

New College, University of Oxford

Doctor of Philosophy (DPhil) Thesis

Michaelmas Term 2015

Abstract

In most organizations, the number of files increases at a rate similar to the growth of data. As one of the big data challenges, many enterprises encounter a common difficulty in a routine operation, that is, finding files in a large-scale file system typically distributed across several physical sites and accessed by thousands of users. This thesis addresses a central question: whether or not visualization techniques can be used to improve the effectiveness and efficiency in performing numerous file searching operations at an industrial scale. All work conducted in this research was done in partnership with Laing O'Rourke as an industrial collaborator.

The main technical approaches to support file searching operations include (a) the use of a database to manage searchable records of files and (b) the use of a search engine to add the exploration of a less-structured file repository. With the rapid increase of files, the former approach incurs a huge cost on entering records of files into the database, while the latter suffers from unreliable search results (false positives and false negatives) and difficulties in collaborative search. This thesis focuses on the second approach, that is, to develop a visualization-assisted enterprise search engine.

In this thesis, we propose two novel visualization techniques in conjunction with an experimental enterprise search engine. The first technique provides users with focus+context visualization of search results (focus) in relation to the search space (context). This assists users in identifying false positives rapidly, and helps users hypothesize potential false negatives and investigate them through the refinement of search criteria. A number of methods for depicting the multivariate information associated to search results

were designed, implemented and compared. Empirical studies were conducted to discover the visual attributes for glyph-based and animation-based methods, and to evaluate different visual designs.

The second technique provides users with support for search activities over a period of time and in collaboration. We developed the novel concept of Search Provenance Graph (SPG), and a method for connecting semantically similar queries in SPGs. Methods and software for visualizing SPGs were designed and implemented, enabling users in collaboration to acquire provenance information efficiently and formulate/reformulate queries effectively.

In conjunction with the research on visualization techniques, we developed an experimental enterprise search engine, which allows visualization components to be integrated. The search engine is knowledge-based, and is supported by multiple ontologies and crawler agents for exploring the search space. We used query-expansion and results ranking to reduce false positives and negatives, active-learning to enable dynamic learning during search operations, and history-based indexing to facilitate real-time return of search results.

This research is the first step towards the development of visualization-assisted enterprise search engines as a new technology that can address a major big data challenge in industry, and can bring a significant amount of cost-effectiveness to everyday operations.

Dedicated to my parents.

Acknowledgements

I owe a lot of thanks and gratitude to many people who have been part of my DPhil journey. First and foremost I would like to express my deepest gratitude to my supervisors Professor Min Chen and Dr René Bañares-Alcántara

To Professor Min Chen, who has guided me through the entire process: without his teaching, guidance, and endless patience, this thesis would not have been possible. It has been an excellent opportunity to work with someone held in such high esteem by the visualization community. I am so fortunate to have been given the chance to do my DPhil under his supervision.

To Dr René Bañares-Alcántara, who has provided his patient guidance, encouragement, and advice during my time as his student: I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly.

I am grateful to Laing O'Rourke for funding my DPhil project, in particular colleagues from Laing O'Rourke: James Eaton, Urszula Kanturska, and Tom Waters, for their collaboration and support during this project. I am also thankful to other Laing O'Rourke colleagues: Patrice Kiell, Nick Lester, Claire Jones, Jasvir Mann, Florian Le Gouriellec, and Tes Adamou for their help and sharing their knowledge during my placement at their company.

I would like to thank my incredibly talented colleagues from Oxford University for their availability to share their knowledge. In particular, I wish to thank Dr Karl Proctor, Dr Simon Walton, Dr Brian Duffy, and Dr Eamonn Maguire. I am grateful to Sohan Sangwan for sharing his knowledge in Web technologies.

Finally, I would like to express my love and gratitude to my parents and my brother Maman Khan for their endless support and inspiring influence throughout.

Thesis Outputs

Publications

The work conducted in this thesis has led to the following publications.

Saiful Khan, Karl J Proctor, Simon Walton, René Bañares-Alcántara, and Min Chen. *A Study on Glyph-based Visualisation with Dense Visual Context*. In *Computer Graphics and Visual Computing*, The Eurographics Association, 73–80, 2014.

Saiful Khan, Urszula Kanturska, Tom Waters, James Eaton, René Bañares-Alcántara, and Min Chen. *Ontology-assisted Provenance Visualization for Supporting Enterprise Search*. In *Advanced Engineering Informatics*, 30(2):244–257, 2016.

Saiful Khan, James Eaton, René Bañares-Alcántara, and Min Chen. *Active Learning of Ontologies for Improving Recall and Precision in Enterprise Search*. Submitted.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Aims and Objectives	5
1.3	Thesis Outline	6
2	Background	9
2.1	Overview	9
2.2	Enterprise Search	10
2.2.1	Databases	10
2.2.1.1	Relational Data Model	10
2.2.1.2	Document-oriented Data Model	12
2.2.1.3	Graph Data Model	13
2.2.2	Desktop Search	14
2.2.3	Personal Information Management	14
2.2.4	Enterprise Search Engine	16
2.2.4.1	Ontology-based Retrieval	17
2.3	Search-related Visualization	18
2.3.1	File System Visualization	23
2.3.1.1	Treemap-based Visualization	24
2.3.1.2	Pixel-based Visualization	26
2.3.2	Multivariate Visualization	27
2.3.2.1	Glyph-based Visualization	28
2.3.2.2	Application of Glyph-based Visualization	30
2.3.3	Provenance Visualization	32

CONTENTS

2.3.3.1	Text-view	32
2.3.3.2	Node-link Graph	33
2.3.3.3	Radial Layout-based Graph	35
2.3.3.4	Other Provenance Visualization Techniques	35
2.4	Building Information Management and Search	36
2.5	Summary	38
3	System Overview	41
3.1	Introduction	41
3.2	Enterprise Data Management and Search	42
3.2.1	Database	42
3.2.2	Enterprise Search Engine	44
3.2.3	Visualization-assisted Enterprise Search Engine	45
3.2.3.1	Why Visualization?	45
3.3	System Overview	47
3.3.1	Search Query Form	48
3.3.2	Query Processing	51
3.3.3	Result Processing	51
3.3.4	Search Result Visualization	52
3.3.5	Ontology Management	53
3.3.6	Active Learning	53
3.3.7	Index Management	54
3.3.8	Crawler	55
3.3.9	SPG Management	55
3.3.10	Search Provenance Visualization	56
3.4	Business Cost Estimation	57
3.4.1	Project Value	57
3.4.2	Life-cycle Cost	60
3.4.3	Search Cost	60
3.5	Summary	61
4	Ontology-assisted Enterprise Search Engine	63
4.1	Introduction	63

4.2	Related Work	65
4.2.1	Enterprise and Desktop Search	65
4.2.2	Ontology-based Retrieval	66
4.2.3	Ontology Learning	67
4.3	System Architecture	68
4.4	Multi-Ontology based Search	70
4.4.1	Ontologies for Knowledge Management	71
4.4.2	Query Expansion	73
4.4.3	Ranking	74
4.4.4	Relevance Feedback	76
4.4.5	Training and Active Learning	78
4.4.6	Weight Normalization	79
4.5	Experimental Results and Discussion	81
4.5.1	Experimental environment.	81
4.5.2	Data Set	81
4.5.3	Experimental Results	84
4.6	Conclusion	89
5	Search Result Visualization	91
5.1	Introduction	91
5.2	Related Work	94
5.2.1	Search-related Visualization	94
5.2.2	Treemaps	96
5.2.3	Glyphs	97
5.2.4	Animation in Visualization	97
5.3	Design Considerations	97
5.3.1	Multivariate Glyph	98
5.3.2	Focus and Context	98
5.3.3	Overview First and Details on Demand	99
5.3.4	Occlusion	99
5.4	Visual Mapping of Glyph	100
5.5	Cyclically Animated Glyph (CAG)	103
5.6	Pre-development Empirical Studies	106

5.6.1	Controlled Experiment One	107
5.6.2	Controlled Experiment Two	109
5.6.3	Controlled Experiment Three	112
5.6.4	General Discussion	113
5.7	Post-development Evaluation	113
5.7.1	Organization	113
5.7.2	Survey Results	114
5.7.3	Summary of Open Discussions	115
5.8	Conclusion	116
6	Search Provenance Graph and its Visualization	119
6.1	Introduction	119
6.2	Related Work	121
6.2.1	Search-related Visualization	121
6.2.2	Provenance Data Visualization	121
6.2.3	Search Provenance or Log Visualization.	122
6.2.4	Ontology-assisted Visualization	123
6.3	Search Provenance Graph (SPG)	123
6.4	Computing Query Similarity for SPG	126
6.4.1	Keyword Similarity	127
6.4.2	Similarity: Type, Size, and Date Fields	128
6.4.3	Aggregate Score	129
6.5	Visualization of SPG	129
6.5.1	Glyphs for Search Queries	130
6.5.2	Glyphs for Search Results	131
6.5.3	An Example Use of the Provenance Visualization	132
6.5.3.1	SPG: Graph View	132
6.5.3.2	SPG: Glyph-based View	133
6.5.3.3	SPG: Collaboration Summary View	136
6.6	Evaluation	138
6.6.1	Requirement Analysis	138
6.6.2	Organization	138
6.6.3	Procedure	139

6.6.4	Discussion	139
6.7	Conclusions	142
7	Conclusion	143
7.1	Summary of Contributions	143
7.2	Summary of Future Work	148
A	Search Index Management	151
A.1	Crawler	151
A.2	Managing an Index	152
A.2.1	Term Processing	152
A.2.2	Index Data Structure	153
A.2.3	Backup and Restore	156
A.2.4	Search	159
A.2.5	Search in a Large Index	159
A.3	Managing a Dynamic Index	160
B	Stimuli For Empirical Studies	165
B.1	Stimuli for Experiments One, Two and Three	165
B.2	Focus Group Study: Questionnaire	165
B.2.1	Static dots vs. animated dots	166
B.2.2	Static dots vs. static glyphs	167
B.2.3	Static glyphs vs. animated glyphs	170
B.2.4	Animated dots vs. animated glyphs	170
B.3	Results of The Questionnaire	171
	Bibliography	185

1

Introduction

Contents

1.1	Motivation	2
1.2	Aims and Objectives	5
1.3	Thesis Outline	6

With a growing amount of data in enterprises, managing and searching for information is becoming a challenge. In Information Retrieval (IR), the concept of *Enterprise Search Engine* was introduced to encourage real-world application of IR, where an enterprise develops an integrated data infrastructure and enables its stakeholders to have an effective search interface for information retrieval.

Grefenstette [93] estimated that the enterprise search industry is growing at 20% per year. A tutorial in ACM SIGIR, 2010 [63] highlighted some of the key challenges in the area of enterprise search. Although these challenges are similar to Web search, many cannot be addressed by existing Web search solutions [63]. For example, the structure of enterprise or desktop data is different from that of Web data. The popular ranking technique (e.g., PageRank [157]) is not effective in enterprise search [100]. Relevance feedback information can be used to learn and improve the rank of search results in the enterprise search [54]. However, such techniques often lead to an extensive amount of search results but do not guarantee adequate recall and precision. In enterprise search, discovering information that are searched and found by experts and colleagues is often part of a search strategy [14, 63]. Systematic support for such collaborative search effort is thus highly desirable.

File search is an elementary task in Enterprise Search. In a construction company, for example, a large number of files are captured and stored in each project.

These files may be different types of reports, CAD files, presentations, scanned copies, spreadsheets, art works, images, audios, videos and so on. It has been reported that stakeholders (e.g., engineers, managers, administrators) spend around 30% of time in file search [5, 96, 97]. However, searching through a large data infrastructure is a time consuming operation, as (i) search queries often return a long list of files, (ii) the list usually includes irrelevant files (false positives), and (iii) the queries often miss some target files (false negatives).

1.1 Motivation

This research was motivated by a significant industrial challenge in the construction industry. As part of Building Information Modeling (BIM) [206, 217], the industry is expected to meet a set of new requirements- (a) for providing clients with the relevant data about a building and the building process at the end of each building project and (b) for monetizing the external value of the digital assets by being able to search and reuse them. While the industry is formulating a detailed specification about what documents to store and in what formats, it has been developing strategies for meeting the requirements in a cost-effective manner.

The collaborative work with a large construction enterprise, Laing O’Rourke, started in January 2012. We have studied the documents about their requirements, and watched videos featuring discussions about these requirements among senior figures in the industry.

To help us appreciate the scale of file search activities, our industrial partner provided us with statistics about several example projects.

- *Education Building Project* — In this £54-million project, a total of 100,047 files were collected on the server. 8,677 files (8.7%) were recorded in a database, e.g, Asite¹, Microsoft Sharepoint². 4 files were issued to the client based on database search.
- *Large Infrastructure Project* — In this £200-million project, a total of 461,711 files were collected on the server. 15,538 files (3.4%) were recorded in a database. 2,626 files were issued to the client based on database search.

¹www.asite.com

²products.office.com/en-GB/sharepoint

- *Office Building Project* — In this £250-million project, a total of 591,906 files were collected on the server. Out of them only 45,000 files (7.6%) were recorded in a database. 6,566 files were issued to the client based on database search.

The above real-world scenarios highlighted several challenging requirements. Firstly, although the database has likely captured most important files, more than 90% files are not searchable through the database. The poor coverage by the database was largely due to the effort required to enter a file record for each file into the database.

We considered three main strategies for addressing these requirements, namely-

- (1) **A Database** requiring each building project to maintain a high quality database of all documents, from which a collection of the desired files can easily be found at the end of the project.
- (2) **An Enterprise Search Engine** requiring all files to be accessible by a search engine, which will be used by a team of administrators to compile a collection of the desired files when they are needed.
- (3) **A Visualization-assisted Enterprise Search Engine** supporting (2) with advanced visualization capabilities.

As part of the overall evaluation of these three strategies, the D.Phil. research project presented in this thesis was formulated to study the feasibility of strategies (2) and (3). In particular, the foci have been placed on two research questions: what technical features of search engine are required, and how visualization can be used to support file search?

It is not difficult to recognize that the construction industry has a large number of stakeholders (e.g., clients, architects, suppliers, local governments, transport authorities) and many teams within each construction enterprise. Many building projects last for years. The files accumulated during each project may be distributed in different parts of a large file system. Each file may have different versions. Each team or stakeholder may have different naming conventions. Some projects may share a certain number of files. Errors in file naming and placement are also expected.

Strategy (1) has shortcomings, to mention a few: (i) the commitments of data archivists for each project and the risk of a single-point of failure within each project and (ii) the cost involved in entering the large volume of files to the database.

All of the above suggested that the *Database* strategy might not be a winning strategy either, though strategy *Enterprise Search Engine* also has shortcomings to name a few:

- (a) **Multivariate Information** The first challenge of strategy (2) is to deal with search results, typically presented in a list or tabular view. Therefore, searching for the relevant files (e.g., based on certain attributes) among a long list of search results, where often most of them are false positives, is a time consuming activity.
- (b) **False Positives and Negatives in Search** The second major challenge of strategy (2) is to deal with *false positives* and *false negatives* in a search process. Given a set of search criteria, a search engine executes a query, returning back a set of results. Among these results, there are true positives and false positives. Meanwhile, among the vast number of files not returned by the search engine there are true negatives and false negatives.
- (c) **Search Collaboration** The third challenge of strategy (2) is that it does not support search collaboration. In an industrial-scale search scenario supporting an effective collaboration among administrators is essential.

The real-world scenarios highlighted several challenging requirements. Firstly, although the database has likely captured most important files, more than 90% of the files are not searchable through the database. The poor coverage by the database was largely due to the effort required to enter a file record for each file into the database. Using the *Office Building Project* as an example, if entering a file record into a database takes 40 minutes on average (including some reading, figuring out metadata, and record maintenance), entering 45,000 files would have taken 30,000 person-hours or 3,750 person-days. If all 591,906 files had to be recorded in the database, it would take nearly 49,325 person-days. Secondly, as part of the new BIM regulation [206], the files to be issued to a client will likely be in thousands if not more. Again, we use the *Office Building Project* as an example. With a database, if one takes 2 minutes on average to find a file in the database to be issued to a client, 6,566 files would take about 219 person-hours or about 27 person-days. Finally, when a search engine was used instead of a database, we do not require any database effort, but the search would take longer time. If one takes 15 minutes on average to find a file using a search engine, 6,566 files would take about 1,642 person-hours or about 205 person-days.

In the context of the enterprise search engine and its application, it is important for the users to be able to identify and pick out true positives quickly among false positives. It is equally, if not more important, for users to recognize the existence of false negatives, and be able to hypothesize where these may be located in the file system and what search criteria may reveal them. It is also important for the users to explore search criteria (i.e., their hypotheses), and to reformulate search criteria progressively.

It is necessary to note that the goal of a user's task is not to formulate the best set of search criteria, but to work efficiently in collaboration to find all required files. Once all the required files have been found, the user will likely stop the process even the best set of search criteria has not been found. In many cases, the best strategy may be to find a complete collection of the required files using a small number of queries even though the search criteria for each query may not be optimal. Meanwhile users' experience of previous projects will play a significant role in formulating search criteria and an effective search strategy. When a search process takes a few days and/or involves more than one administrator, effective visualization and interaction support for search and provenance is essential.

1.2 Aims and Objectives

The overall aim of this research is to provide industry a visualization assisted enterprise search technology that can address the problems emerging from handling large number of files in a cost effective way.

We focus on several major objectives to achieve it:

1. To investigate how to overlay search results on top of a search space in the context of large-scale distributed file systems to facilitate focus+context visualization.
2. To develop a search provenance management technique and to introduce provenance data visualization techniques.
3. To develop a special purpose enterprise search engine for enabling the integration of visualization facilities which cannot be done by a general purpose or open source enterprise search engine.
4. To investigate how machine learning techniques can be deployed in an enterprise search engine to improve the precision and recall of search results.

5. To evaluate all the techniques developed in this project in conjunction with the industrial partner.

1.3 Thesis Outline

Now that the main challenges and the potential solutions for industrial scale file management and search problem have been explained more detailed content of this thesis can be given. The thesis is organized into five chapters. Chapter 2 will present the review into the current state-of-the-art of file search, ontology-assisted search, and visualization techniques to support the search result and search provenance. Chapter 3 will present an overview of the system and the business cost advantage of visualization-assisted search. Chapter 4 will give the details of the prototype enterprise search engine, enabling the experimentation with visualization facilities in later chapters. Chapter 5 and 6 will aim to answer how visualization can assist in enterprise search before conclusions are presented in Chapter 7.

Chapter 2: The Background of this work mainly builds on two areas (a) enterprise search engine, and (b) visualization related to search and search history or provenance. Firstly, we will present the current literature on enterprise and desktop search, knowledge-based or ontology-based retrieval, and ontology learning. Secondly, we will present the surveys in the areas of visualization related to search, file system visualization, and provenance visualization. Finally, a survey on the off-the-shelf BIM software used in construction industries will be presented in this chapter.

Chapter 3: System Overview will present an overview of the system, e.g., search query and result processing, visualization of search results and search provenance, active learning using user feedback information, distributed system crawling and index management process using a flowchart. This chapter will also discuss a business cost estimation to analyse the cost benefit of visualization-assisted enterprise search over other alternatives solutions, as stated in Objective 5.

Chapter 4: Ontology-assisted Enterprise Search Engine will present a number of technical challenges associated with the development of a scalable search engine for an enterprise. An architecture of a special purpose enterprise search engine and its functional modules to facilitate the development and experimentation of

the visualization facilities (in the following chapters) will be discussed first. Thereafter, we will discuss multiple ontologies to provide a knowledge-based support for query enrichment and ranking, followed by relevance feedback, processing and active learning of the ontologies. Finally, the combination of software prototyping and scalability testing will be presented to investigate the feasibility of using ontology-based ranking and active learning of ontologies for large-scale file operations in a real-world application. This chapter will address Objectives 3 and 4.

Chapter 5: Search Result Visualization will focus on the visualization of the search results and their distribution in the search space to help administrators perform a quick visual search of the search results and for identifying the false negatives and false positives in a search. Thereafter, this chapter will present empirical studies to be conducted to improve a glyph representation on top of a treemap. This chapter will address Objective 1.

Chapter 6: Search Provenance Graph and its Visualization will focus on the visualization of the search provenance information. The provenance information can assist the administrator to learn from their historical queries and to formulate/reformulate search queries efficiently, perform a search task collaboratively, and to identifying false positives in a search. This chapter will introduce a graph structure called Search Provenance Graph (SPG) for recording the search provenance in individual projects. The SPG will be visualized with a combined glyph-graph visual representation as well as a graph view and a collaboration summary view. This chapter will address Objective 2.

Chapter 7: Conclusion will summarize the thesis with a discussion on its contributions and future research directions.

Appendix A: Search Index Management will present the data structure used by the enterprise search engine for index management. The main operations and algorithms used for index management will be described in this Appendix.

Appendix B: Stimuli for Empirical Study will illustrate the stimuli used in empirical studies to be conducted in Chapter 5. The questionnaires and the results of the focus group discussion around four different glyph design options will also be presented in this Appendix.

2

Background

Contents

2.1 Overview	9
2.2 Enterprise Search	10
2.3 Search-related Visualization	18
2.4 Building Information Management and Search	36
2.5 Summary	38

2.1 Overview

This chapter introduces the work related to this thesis: enterprise data management and search, visualization related to search, and off-the-shelf BIM software and tools. The first section provides a background of the industrial scale file management and search challenges and efforts made to address them. In particular we survey how ontologies have been widely used as a knowledge base to support domain-specific information retrieval. The second section introduces the visualization techniques related to search: display of search attributes. In this section, we also discuss provenance information visualization techniques. Finally, it is also essential to review and consider the existing solutions designed for file management and search used by construction companies.

2.2 Enterprise Search

File or document search is an elementary task of enterprise search. There are numerous solutions that have been developed to store a large quantity of data generated in a big company and to make them efficiently searchable. This area can be classified and broken down into four main categories: (a) Databases, (b) Desktop Search, (c) Personal Information Management, and (d) Enterprise Search Engine.

Databases are widely used as a data management and search system tool used to systematically store and access documents using various data models. Desktop search mainly focuses on searching for files stored on the hard drive of a computer. Personal Information Management extends the desktop search work by providing a data model to store and access documents. Enterprise search engine deals with searching for files as well as searching for other relevant information in an integrated data infrastructure. It enables access to all the information stored in a company, e.g., desktops, file systems, servers, databases, email servers, and so on.

2.2.1 Databases

Relational databases are the most popular data management system used to store structured data in a relational data model. However, at present in addition to structured data a semi-structured and unstructured data are being generated with greater velocity. As a result, there have been numerous attempts to create different data models, such as the document-oriented data model, graph data model, key-value store which are among the most popular.

2.2.1.1 Relational Data Model

The relational data model uses a predefined blueprint for the tables or schema defining the relationships between tables of data, as shown in Figure 2.1. Within a table, named columns, as well as the type of data that can be stored in each column, are defined. The central concept in the relational data model is that similar kinds of data is stored in the rows of the same table with a unique primary key. In instances where two tables are related to each other, a field is selected to group them together, a foreign key.

A Data Definition Language (DDL) is used to define the different structures in a database, and the common DDL statements are “create”, “alter” and “drop”. The

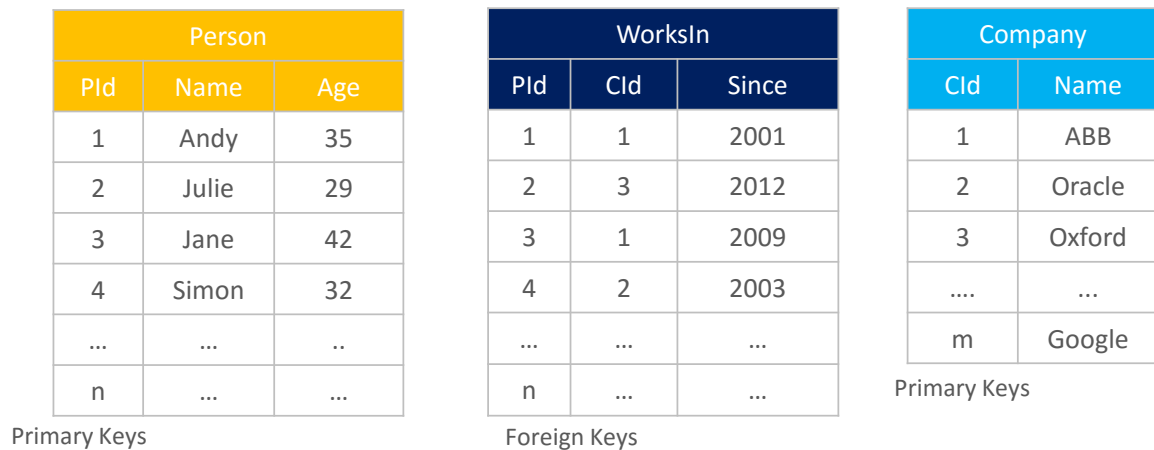


Figure 2.1: An example relational data model. Person and companies details are stored in a Person and a Company table. A WorksIn table contains the relationships between person data and company data.

data attributes are added, updated, and selected in tables by the administrators using a Data Modification Language (DML), e.g., Structured Query Language (SQL). The relational model is simple and highly scalable. There are several commercial relational databases: MySQL¹, Oracle², and PostgreSQL³ to name a few. Commercial relational database-based tools for document management, e.g., Microsoft Sharepoint⁴ are commonly used in various enterprises, and the Asite⁵ system was designed specifically for construction document management and search.

Figure 2.1 demonstrates a naive example where records of people, companies they work for, and for how long they have been working there are stored in three different tables.

In a relational model, as the records are stored in a predefined schema or format, changing that schema is very expensive. For example, if in the future we require additional data attributes to be stored e.g., people's social security number and/or company address then we need to create a new schema. Moreover, when new and complex forms of relations start to emerge (for example, previous employment history details/or dates in relation to) it would be difficult to model and migrate the old data to a new schema relational model.

At present, in addition to structured data, semi-structured and unstructured

¹www.mysql.com

²www.oracle.com/database/index.html

³www.postgresql.org

⁴<https://products.office.com/en-us/sharepoint/collaboration>

⁵www.asite.com

2. BACKGROUND

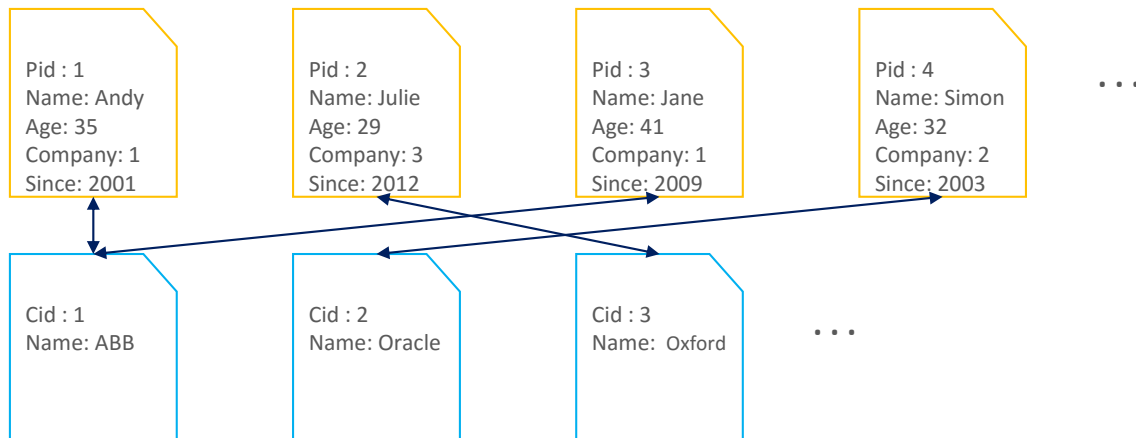


Figure 2.2: An example document data model. Instead of splitting objects into tables and rows, they are represented in two different document structures one for persons and one for a company. By providing a reference in the person document to a company document, we have created a relationship between the two entities.

data are being generated with greater velocity. As a result, there have been numerous attempts and efforts to create different data models that can eliminate the need for a predefined blueprint or schema. Among them document-oriented data model, graph data model, and key-value store are the most popular.

2.2.1.2 Document-oriented Data Model

Due to the limitation of the need for a predefined blueprint or schema to store data in a relational data model, document-orientated databases were developed. Such databases use a document-orientated data model that follows no internal structure, i.e., the fields and relations do not exist as predefined concepts. It acquires the type information from the data itself. Instead, all of the data attributes for an object are placed in a single document, and are stored as a single entry. A relationship between the two entities is created using references.

Normally all related information is stored together and it allows every instance of data to be different from any other. Thereby, creating a data model that is flexible to future changes. For example, if in the future we require the storing of additional data attributes, types, and relationships these can be easily mapped into program objects. CouchDB⁶ and MongoDB⁷ are the two most popular document-oriented

⁶couchdb.apache.org

⁷www.mongodb.org

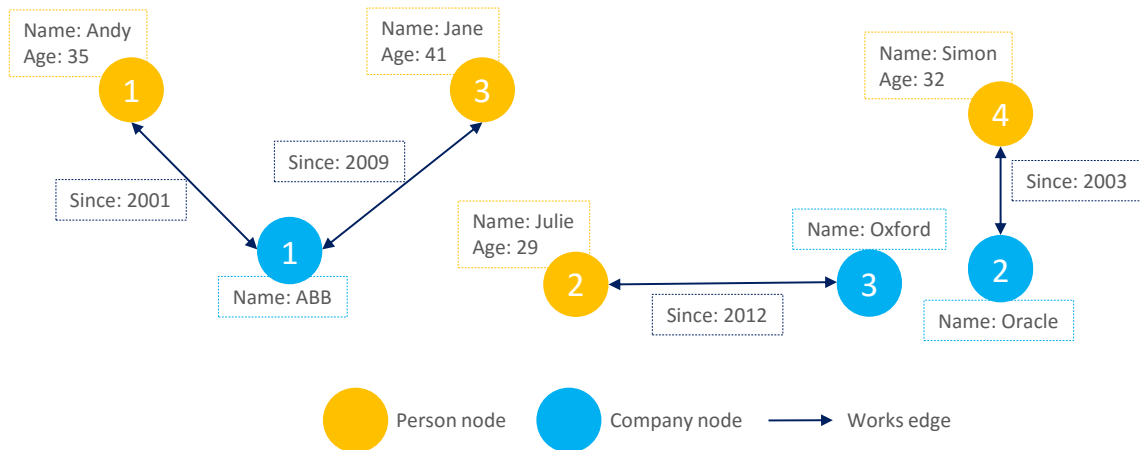


Figure 2.3: An example of a graph data model. Here, each instance of people and company data are represented as nodes, and edges between the nodes represent the relationships between the nodes. As many attributes as required could be added to both edges and nodes. A directed edge has an arrow pointing to its head node while an undirected edge does not have any arrows.

databases. XML databases are a kind of document-oriented databases that are optimized to extract their metadata from XML documents.

In the people and company example, the document would contain the people's name, age, working since, and company name in a single record, see Figure 2.2. In future, if we want to store additional data attributes, e.g., people's social security numbers and/or company addresses then unlike in the relational data model, we would not need alter the existing data.

2.2.1.3 Graph Data Model

Graph databases have received much attention and interest in recent years for exploring data that is structured as a graph (or tree), in particular when the relationships between data points are significant. Graph databases implement a graph data model that are pairwise relations between objects using "vertices" or "nodes" and "edges" connecting them. All the nodes and edges can have a type (a label) to classify them. Both nodes and edges can have one or more properties or attributes. Like the document-oriented data model, the graph data model also gets the type information from the data itself that allows every instance of data to be different from any other. The graph data model can represent richer and complex relation-

ship using edges and properties. Popular graph databases are Sparksee⁸, Neo4j⁹, OrientDB¹⁰, InfiniteGraph¹¹, and AllegroGraph¹² to name a few.

In the people and company example, the people nodes contain the people's name and age, and the company nodes contain the company name. The edges, "working since", highlight the relationship between people and company, see Figure 2.3. Such a data model is flexible to future changes and updates, and can also represent more complex data and relationships.

Although the documented-oriented and the graph data model eliminate the need for a future-proof complex schema to store and update data, the resources needed to enter records into a databases is costly and expensive, particularly when the scale of the data is large.

2.2.2 Desktop Search

GNU/Linux utilities *find* and *grep* are the simplest command line based file search tools. Given a search term as an input, these tools scan through the location of a disk drive and return all files matching the term. GNU/Linux utility *locate* speed up this process by using a pre-built database, however, like *find* and *grep* it cannot filter results by file attributes.

Recently there have been numerous commercial tools developed for improving file search on desktops, e.g., Microsoft Windows Desktop Search, Google Desktop, Apple MAC OSX Spotlight, and Zeitgeist on Ubuntu GNU/Linux. Keyword-based search for the discovery of software was proposed in [61, 158, 159].

2.2.3 Personal Information Management

The work on Personal Information Management (PIM) system aims at providing users with generic and flexible data models to access and store information, this is beyond what is supported by the traditional file system search tools. The SEMEX [35] and Haystack [114] tools implemented an enhanced search capability, allowing users to organize their data semantically by creating associations between files or data entities and then leveraging these associations during search. The

⁸www.sparsity-technologies.com

⁹neo4j.com

¹⁰orientdb.com/orientdb

¹¹www.objectivity.com/products/infinitegraph

¹²franz.com/agraph/allegrograph

iMeMex Data Model (iDM) [62] was proposed to represent unstructured, semi-structured as well as structured data within a single model. Xu *et al.* [226] proposed a generic data model to capture and represent file semantics. They suggested a framework of basic relations and tools for generating and using semantic metadata.

A small number of research papers have focused on querying metadata fields such as file size, date, or types as filtering conditions to improve search capabilities. Khoo *et al.* [120] showed that desktop users tended to prefer location-based search to keyword-based search. They also reported that users rely on metadata information such as directories, dates, and names when searching for relevant files. Teevan *et al.* [199] investigated the users' behaviour when searching for e-mails, files, and the web. They found that even if users knew exactly what they were looking for, instead of keyword-based search, they often navigated to their target in small steps using contextual information, e.g., metadata information. The latest work on PIM [211] implemented a multidimensional search approach that allows users to perform fuzzy searches for structure and metadata conditions in addition to keyword conditions.

For semi-structured document retrieval, researchers have adapted traditional retrieval models to handle documents with multiple fields. Early works treated each field as a smaller document and combined field-level scores using a linear combination of probabilistic models [155]. The straightforward combination of field-level scores have several limitations, this has led to efforts such as BM25F [174]. Several works focused on improving the quality of document search systems. "Stuff I've Seen" [74] given importance to file date since most users appear to sort the results by date. This system facilitates information re-use by finding and re-using previously searched information. The desktop and PIM search provide limited features to support interactive user interaction. Phlat [55] developed an intuitive search interface that allows browsing through a variety of associative and contextual information, and tagging scheme for organizing personal content across storage systems. Shah *et al.* [186] improved the quality of search by introducing the search context [196].

Thomas [201] treated desktop search as meta-search problem for effective retrieval of individual collections. An adaptation of score combination and smoothing was suggested by Zhao and Callan [234] for the language modelling approach to information retrieval. Cohen *et al.* [54] improved the desktop search results by learning feature weights with training data.

Desktop search and PIM tools are mainly designed to search for files or information stored in a limited information space and therefore may not scale up well to a large-scale distributed file system search.

2.2.4 Enterprise Search Engine

The major challenge with Desktop or PIM systems is that they were developed for file management and search in a desktop scale. Such systems do not scale well to a large-scale distributed system search.

Numerous open source enterprise search engines (e.g., Lucene¹³, Solr¹⁴, Elastic-Search¹⁵, Xapian¹⁶, LucidWorks¹⁷, Indri¹⁸, Terrier¹⁹ and so on) were developed for text-based document retrieval. They primarily use Lucene index at the backend. Lucene provides search index (also known as “inverted index”) creation, storage and management facilities with document ranking algorithms (e.g., Boolean, TF-IDF, Cosine, Fuzzy and so on). These modern tools also use probabilistic models to map documents to terms and then rank results. Probabilities are generated using methods such as TF-IDF and other language models. Although the exact methods of commercial systems are unpublished, it is likely they use techniques similar to Lucene and Indri.

Previous research was focused on different mechanisms to improve the search time by enhancing the retrieval capability of enterprise search engines for engineering documents [96, 97], multi-topic documents [130], patents [142], mechatronic data [76] and so on.

However, recent literature has focused mainly upon improving the retrieval capability of an enterprise search engine by knowledge-based support with ontologies for improved query enrichment and ranking, this is discussed further in the next subsection.

¹³lucene.apache.org

¹⁴lucene.apache.org/solr

¹⁵www.elastic.co

¹⁶xapian.org

¹⁷lucidworks.com

¹⁸www.lemurproject.org/indri

¹⁹terrier.org

2.2.4.1 Ontology-based Retrieval

Ontologies have been widely used in domain-specific information retrieval. The Beagle++ [52] search engine was developed for Semantic Desktops. It indexes RDF metadata annotations to represent the meaning of documents together with the document content. The ranking of documents according to an ontology was reported in [41, 123]. They presented an extension of the vector space model [182] and together with document content, they index semantic annotations of documents and use this information for search.

All three promising approaches extend the vector space model using semantic information, however, none of them is able to apply measures of semantic association. They define the annotations for the documents manually. Therefore, Billig *et al.* [20] chose to align the representation of the document to the ontology automatically, using simple string matching.

Vallet *et al.* [207] used ontologies to improve search over large document repositories. This retrieval model is based on an adaptation of the classic vector-space model, including an annotation weighting algorithm, and a ranking algorithm. Other work on file systems, e.g., [44, 95] enhanced the quality of file system search by leveraging the context of the search space. Castell *et al.* [41] proposed a retrieval model based on an adaptation of the classic vector-space model, including an annotation weighting algorithm, and a ranking algorithm. This model builds well-defined ontologies by populating knowledge bases and mapping keywords to concepts. Fernandez *et al.* [81] proposed an ontology-based information retrieval model by exploitation of domain knowledge bases to support semantic search capabilities in large document collection from a Web environment. ReFinder [59] is a context-based information re-finding system which assists users in re-finding their previously accessed files and web pages based on contexts such as time and location. A knowledge-based framework for integrating ontology-based personalized retrieval and reminiscence support was proposed in [189].

Leite and Ricarte [129] proposed multiple related ontologies for knowledge representation and integrated the system with a fuzzy model. A multi-ontology based multimedia annotation model was described in [65]. In this work, a domain independent multimedia ontology was integrated with domain ontologies to provide domain-specific views of multimedia content. Vaidurya [144] is a document search system developed for clinical document search by using a domain specific ontology. A semantic ontology-based framework for personal information management

was proposed in [225]. Moreover, query expansion techniques using ontologies can be found in [19, 60, 151].

Ontology Learning. Ontology learning techniques have been widely studied. A comprehensive review and discussion of major issues, challenges, and opportunities in ontology learning can be found in Zhou’s survey [235]. Ichise *et al.* [108] proposed a framework for organizing the ontology mapping problem into a standard machine learning framework. This framework would use multiple concept-similarity measures. Maryam *et al.* [139] described the bottleneck of ontology learning while dealing with the knowledge acquisition and modelling domain knowledge. They presented a survey of the different approaches in ontology learning from semi-structured and unstructured data. Idrissi *et al.* [77] proposed a solution to overcome the manual knowledge acquisition bottleneck in ontology learning. They also presented a practical study of the methods that take data models as input to the learning process. Zhang *et al.* [230] proposed a semantic retrieval model of engineering domain knowledge, e.g., product design and process planning. They addressed the problems with existing keyword-based and semantic-enabled methods, and proposed an ontology-based semantic retrieval scheme for knowledge search and retrieval from domain documents.

The content-based retrieval systems have certain limitations. In an enterprise, most of the files are non-textual (e.g., CAD, media, archive, images of building designs or construction sites, scanned copies of invoices and receipts, etc.). Therefore, improved metadata-based retrieval is essential. Similar to Web search, an enterprise search engine can use relevance feedback information [63] that can be used to learn and improve the ranking of the search results in the enterprise search [54]. Most importantly, it would be useful to assist search with effective visualization and interaction [17].

2.3 Search-related Visualization

Our work builds on research in the areas of search-related visualization mainly search result visualization and search provenance or history visualization. The work in the area of search-related visualization mainly focuses on the Web-search domain. However, there are no serious effort on assisting industrial-scale enterprise search with visualization.

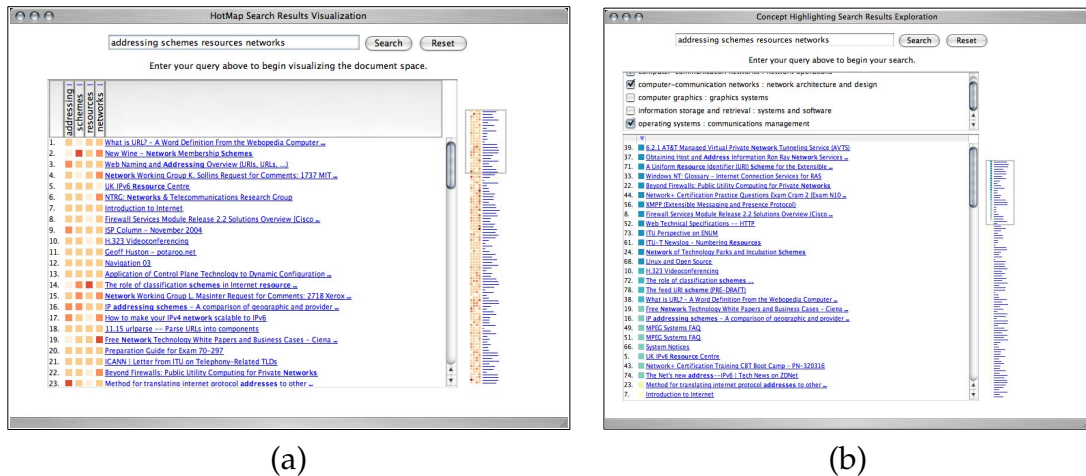


Figure 2.4: (a) HotMap: the term-frequency (tf) of each keyword in the query is shown using colour coding. (b) Concept Highlighter: Firstly a set of concepts related to the search query is displayed, and based on the user's selection the search results are sorted. The sorting is based on a fuzzy membership score and highlighted using colour coding. Adopted from [105].

xFind [7] was developed to explore document search results as a ranked list as well as a scatter plot. In this tool, documents are displayed in a scatterplot where the search attributes, e.g., relevance, file size, and date are mapped to x -axis and y -axis. The attributes can also be mapped to colour and size of the icon. For example, the y -axis can correspond to the modification date of a document, and the document relevance can be mapped to icon size and/or icon colour (e.g., highly relevant in orange, and less relevant in white). Roberts *et al.* used glyphs to visualize web search results [172] where the domains (e.g., .com, .edu, .net, .gov etc.) were mapped to the shape of a glyph.

In Web search, the search results are listed in 10–15 results per page and most users typically only scan the first page [194]. Such list-based representations limit the user to explore most of the search results. Therefore, the HotMap and Concept Highlighter were proposed by Hoeber and Xue [105] to support visual exploration of the Web search results, see Figure 2.4. This tool shows search results in two levels of detail: an overview of the top 100 documents, and a detailed view of 20–25 documents at a time. This work also discussed how these views support the visual query of Web search results. Chau [42] proposed a glyph-based on a flower metaphor to visualize web search results.

Among all of the scholarly work that has been published by the visualization and IR communities on navigation and exploration of Web information space, the

2. BACKGROUND

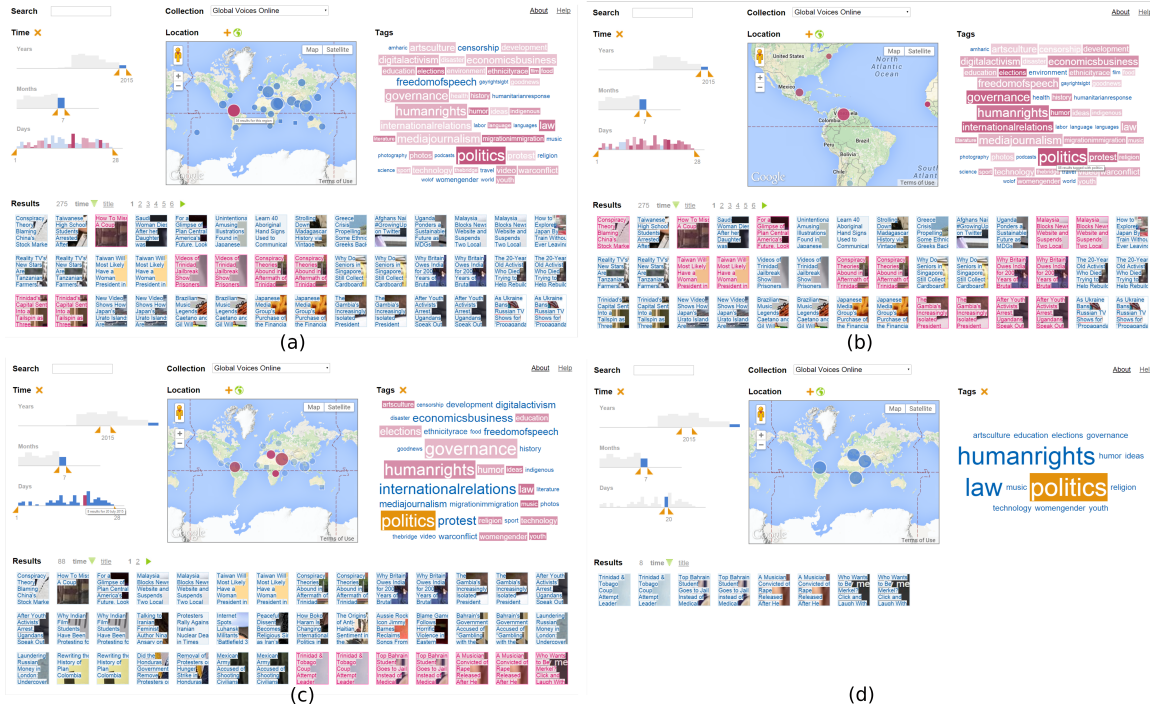


Figure 2.5: Gradual refinement of query parameters in VisGets using the dimension: (a) spatial– the region close to Caribbean is selected, (b) topical– the larger tag “politics” is selected, (c) and (d) temporal– first the month February is selected and then few days around the middle of February are brushed. Adopted from [66].

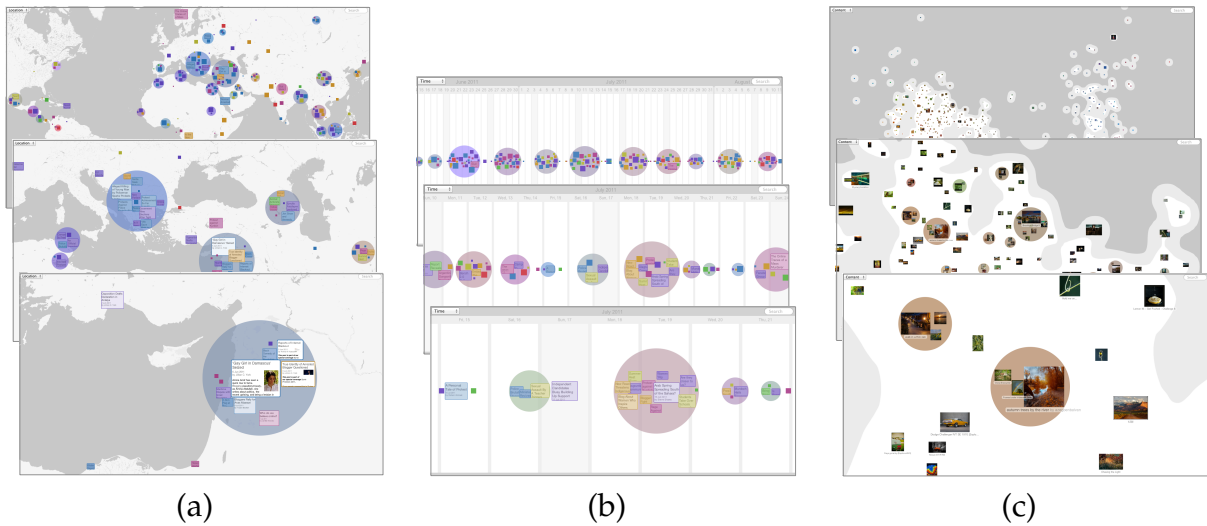


Figure 2.6: Fluid Views: (a) Zooming increases geographical detail in the map, here, from a continental view to the Mediterranean. (b) The time scale is changed from months, to weeks, and days. (c) Base layer gets gradually magnified and the items, filtered and displayed with more detail. Adopted from [67].

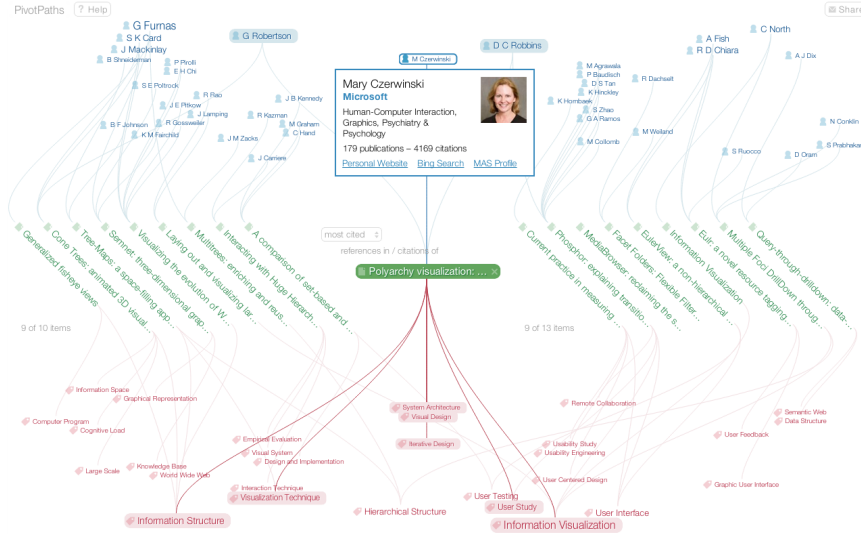


Figure 2.7: PivotPaths tool shows variety of relation types across various dimensions. Adopted from [68].

VisGets [66], Fluid Views [67], and PivotPaths [68] are the most significant. VisGets [66] was developed to visualize different Web search attributes to assist in formulating search queries and to visualize search results using interactive geographic maps and tag clouds. Applications of VisGets to a variety of Web data collections were demonstrated in [69]. An empirical study on interactive and visual exploration of the Web using VisGets was reported in [70], see Figure 2.5. Fluid Views [67] enables users to view Web search results geographically, temporally, and content-wise in dual layers: for overview and details, see Figure 2.6. PivotPaths [68] allows users to explore datasets with a variety of relation types across various dimensions, see Figure 2.7.

Another example of a notable work using Voronoi treemap to organize search results was proposed in [153], see Figure 2.8. ProjSnippet [91] visualizes web search results in a global view and provides a clustering technique for identifying similar content, see Figure 2.9. The similarity measurement used in this work is based on the vector space model. The recent visualization supported tool for document search, Footprints [109], was developed for analysts searching for documents, Figure 2.10.

The visualization techniques in the field of search primarily used geographical maps. In an enterprise, files are scattered across different countries, regions, and projects. Therefore, it would be desirable to show the hierarchical and distributed file system, e.g., a tree-based visualization. File search results consist of multiple

2. BACKGROUND

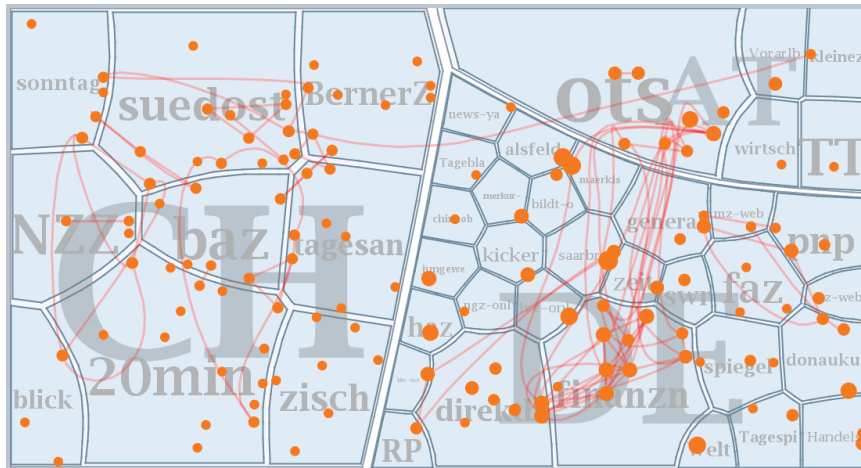


Figure 2.8: Search results are displayed with Voronoi treemap. Adopted from [153].

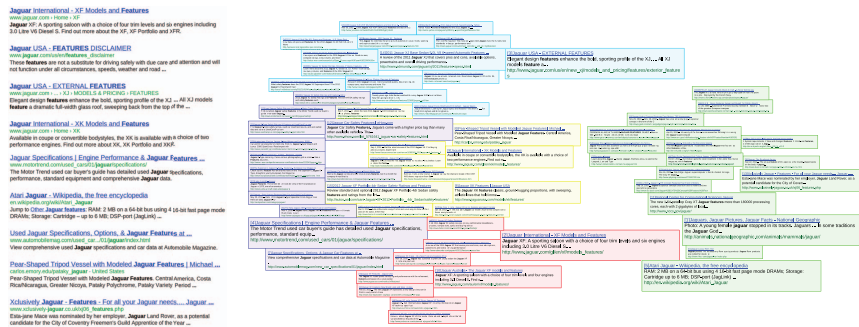


Figure 2.9: ProjSnippet tool visualizes the Web search result and clusters them based on content similarity. Adopted from [91].

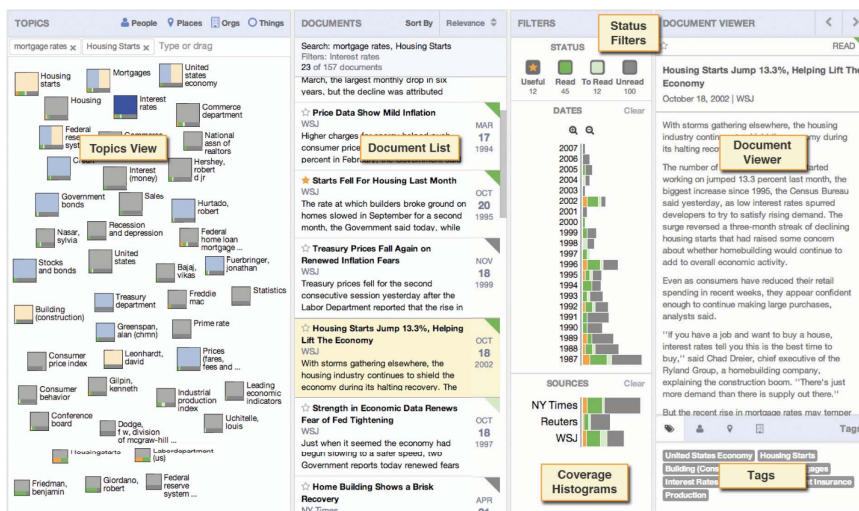
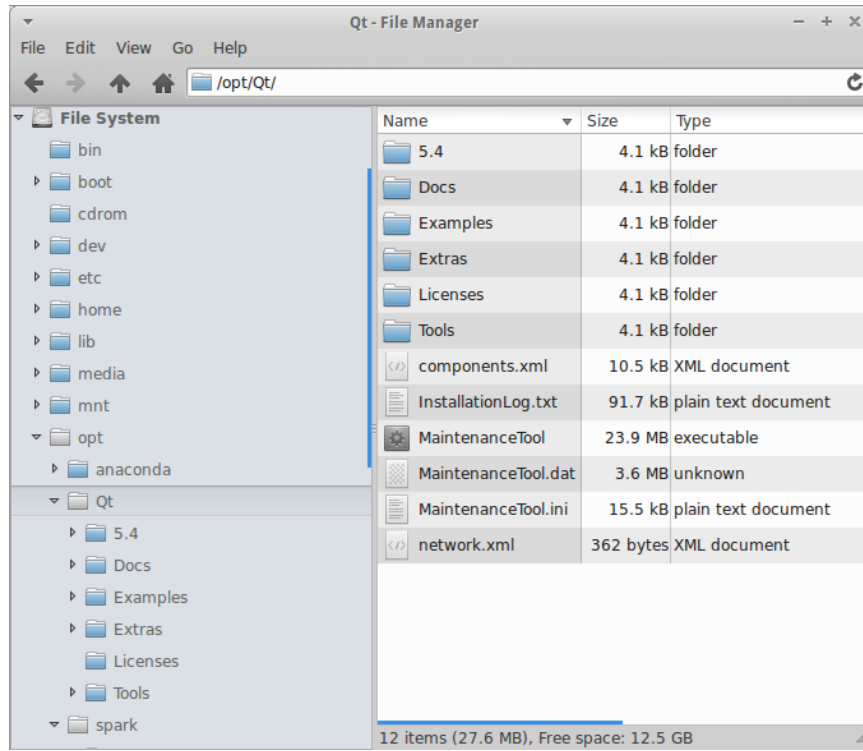
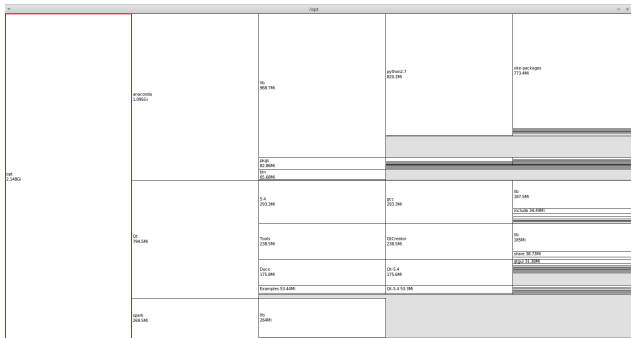


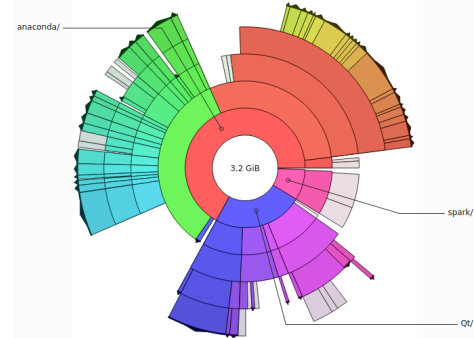
Figure 2.10: Footprints, a document search and visualization tool. Adopted from [109].



(a)



(b)



(c)

Figure 2.11: “/opt” directory of an example GNU/Linux system and its contents are visualized with (a) File System Explorer (b) xdiskusage, and (c) Disk Usage Analyzer.

attributes, therefore multivariate visualization using glyphs would be useful.

2.3.1 File System Visualization

The file systems visualization problem has been widely studied and considered by the visualization community. There are numerous file system visualization-assisted tools and techniques which have been developed for analyzing disk us-

age, browsing hierarchy, searching for files and directories, and so on. The file system is a hierarchical structure, therefore, this essentially falls into the category of a hierarchy and/or tree visualization problem. A commonly used visualization technique for hierarchies is a tree or none-link diagram. For example, in all modern operating systems, the file system Explorer is the primary interface facilitating file system browsing and other functionalities. The Explorer displays the file system hierarchy using a tree as well as file/folder list/tabular view, see Figure 2.11 (a). Such explicit hierarchy visualization techniques do not utilize the space efficiently, therefore, fail to scale well for large file systems. Implicit hierarchy visualization techniques were proposed to solve this problem, a comprehensive survey on such visualization can be found in [185].

The `xdiskusage`²⁰ tool displays file system hierarchy in a simple manner. It shows the hierarchy from left to right with a rectangular shape of size representing its file size. In Figure 2.11 (b), a “/opt” directory of GNU/Linux system is shown as an example. Here, the left most rectangle represents the “/opt” folder while the subsequent rectangles to the right represent the folders down the “/opt” directory. Any folders or rectangle can be zoomed in/out by double-clicking on it.

Baobab²¹, Filelight²², and Philesight²³ create an interactive map of concentric, segmented rings that help visualize disk usage on a desktop computer Baobab is also known as Disk Usage Analyzer and at present is shipped with most GNU/Linux distributions as a default disk usage viewer application. This tool recursively scans the folder defined by a user and visualizes the space taken up by each file and folder under it using a circular rings chart representation. In Figure 2.11 (c), the “/opt” folder was visualized as an example.

2.3.1.1 Treemap-based Visualization

Treemaps were proposed by Johnson and Shneiderman in 1991 in their seminal paper [111,190]. Their objective behind this work was to find out how and where the space of a hard drive is used by multiple users and to find large files that could potentially be deleted. Treemap design was initially improved by Turo *et al.* [204] to display 5000 files. A user study found that novices could see around 20-50 nodes at 1-3 levels, and that zooming through multiple level caused a loss

²⁰xdiskusage.sourceforge.net

²¹www.marzocca.net/linux/baobab/index.html

²²methyblue.com/filelight

²³zevv.nl/play/code/philesight

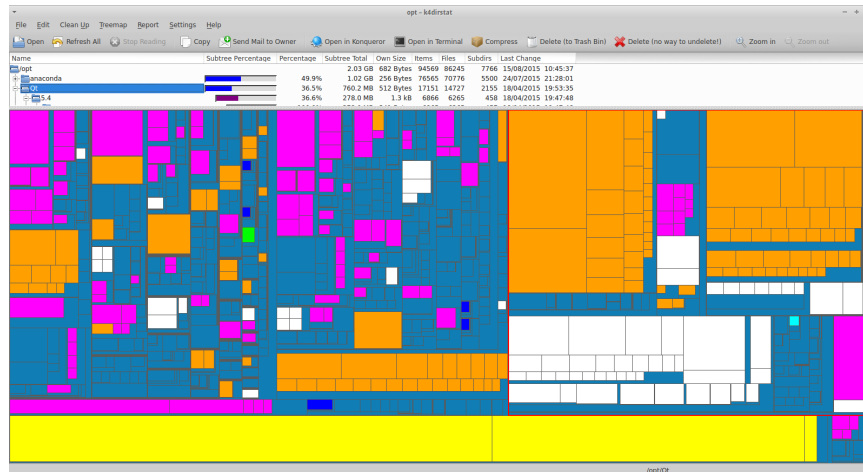


Figure 2.12: Screenshot of k4dirstat tool visualizing the “opt” directory (was also visualized in Figure 2.11) of a GNU/Linux file system using a treemap.

of context. Over the years treemaps have proved to be the most cost-effective and space-efficient graphical technique for visualising hierarchical relationship in datasets [185]. Without going into too much detail regarding different applications of treemaps, in this section we mainly describe different treemap-based tools and techniques proposed for file system visualization. An up-to-date information on various treemaps and their applications can be found in Ben Shneiderman’s Web page [192].

Cushion treemaps by Van Wijk and Van de Wetering [208] apply a texture to the treemap rectangles to make them appear like shiny cushions. This has improved the perception between nodes and eliminated the need for lines to separate the nodes, thereby saving space. Windirstat²⁴ and K4dirstat²⁵ are the most widely used treemap applications developed for file system, see Figure 2.12. One interesting feature of these tools are that they use Cushion treemaps

Lu and Fogarty [131] introduced Cascaded treemaps to improve visibility of nodes. In Cascaded treemaps, a mild 3-D effect is applied to the nodes to improve the visual search in time-critical situations, e.g., real-time stock performance analysis. Zoomable treemaps improved the interaction and navigation capability of treemaps and were found to be effective in exploring large datasets [24]. Zoomable user interfaces facilitates the navigation in large hierarchical datasets, although zooming through multiple levels causes loss of context.

²⁴windirstat.info

²⁵apps.ubuntu.com/cat/applications/k4dirstat/

User evaluation of treemaps has suggested that novices were often struggling to understand the layout that might show 5000 or more files at 7 or more levels [192]. Therefore, to improve the performance of treemaps in terms of order, aspect ratios, and stability, different layout algorithms (e.g., BinaryTree, Ordered, SliceAndDice, Squarified, and Strip) were proposed [192]. Other variations of treemaps using non-rectangular layouts have also been proposed to improve the space utilization.

Different treemaps with non-rectangular layouts are Jigsaw Treemaps [218] (Figure 2.13 (a)), Convex treemaps [156] (Figure 2.13 (b)), Voronoi treemaps [15] (Figure 2.13 (c)). Balzer *et al.* [15] proposed a Voronoi treemap to improve the aspect ratio and increase space utilization by effective use of space by introducing various shaped polygons. Wetzel *et al.* [220] proposed Circular treemaps for effective use of space, see (Figure 2.13 (d)). Auber *et al.* [10] proposed GosperMap in order to assist users to remember the global structure of the hierarchical data, see Figure 2.13 (e).

A visual representation of NTFS file system permission was presented in [103, 171]. They used treemap-based visualization to understand and manipulate file system permissions intuitively.

2.3.1.2 Pixel-based Visualization

Treemaps and pixel-based visualization share a similar dense appearance. Moreover pixel-based visualizations can also be regarded as a special case of a treemap. Keim provided a comprehensive survey of pixel-based visualization [118]. Oelke *et al.* proposed a number of methods for boosting some focus information in a pixel-based visualization [154], see Figure 2.14.

After reviewing different information visualization techniques we have found that visualization of large-scale multivariate file system data is a particular challenge. Treemaps are a commonly used space efficient visualization tool for file system visualization. However, treemaps do not support document search nor are they able to display multivariate file system information. Therefore, how display search attributes: queries, results, and how to effectively present multidimensional attributes per node is a challenge.

Treemaps can only show the structure, however, with multivariate data new methods come into play, see Kerren *et al.* [119].

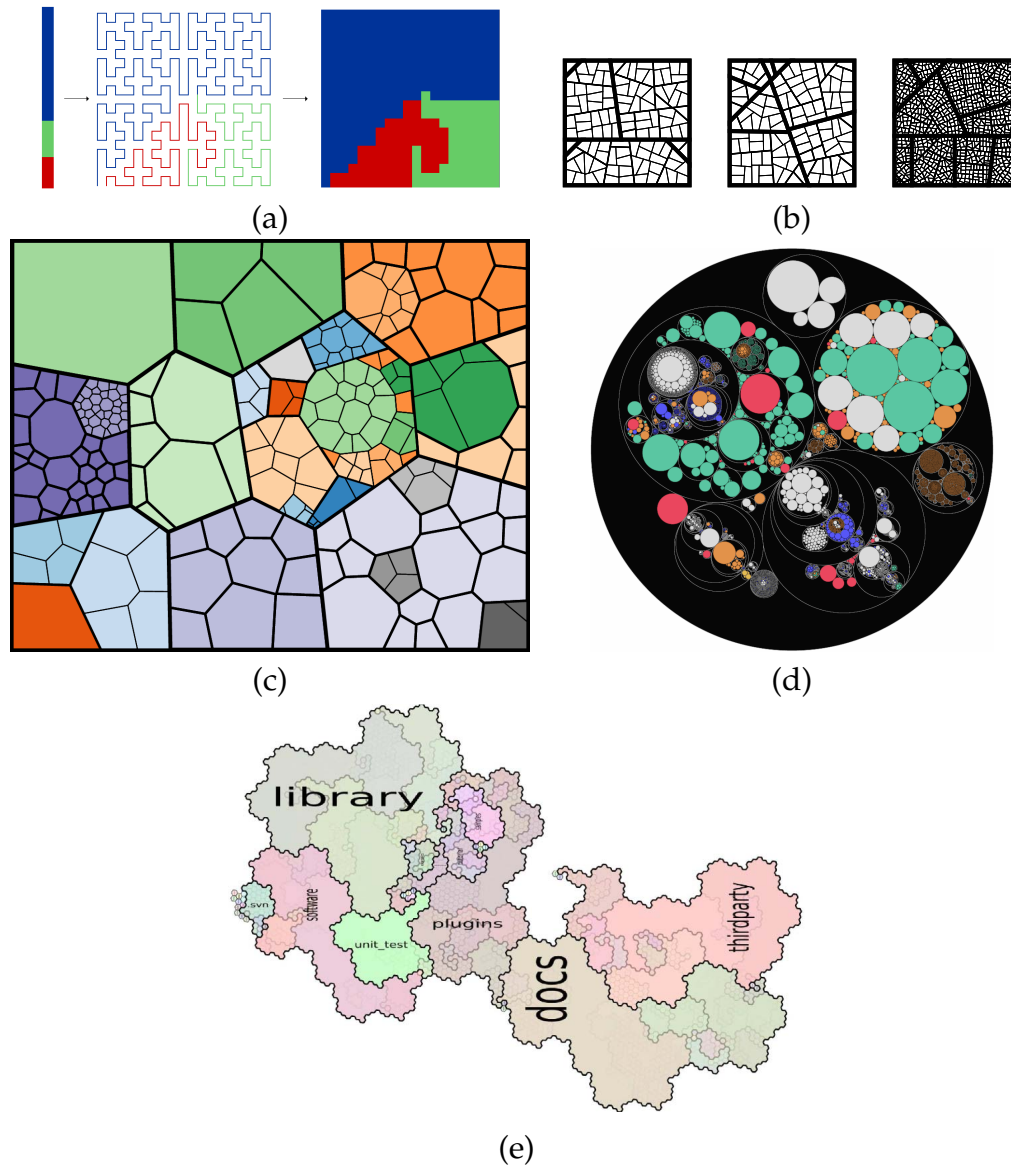


Figure 2.13: Different treemaps with non-rectangular layout: (a) Jigsaw [218], (b) Convex [156], (c) Voronoi [15], created from cse512-14w.github.io/fp-plvines-djpeter/demo.html, (d) Circular lip.sourceforge.net/ctreemap.html, (e) Gosper [10].

2.3.2 Multivariate Visualization

Multivariate information visualization techniques deal with displaying all the data attributes at once. Four main multivariate data visualization techniques are (a) tables, (b) scatter plots, (c) parallel coordinate plots, shown in Figure 2.15, and (d) glyphs. Tables are not scalable in terms of number of rows and columns it can show at a time. It is hard to see patterns, trends, and distribution in a table. Scatter plots

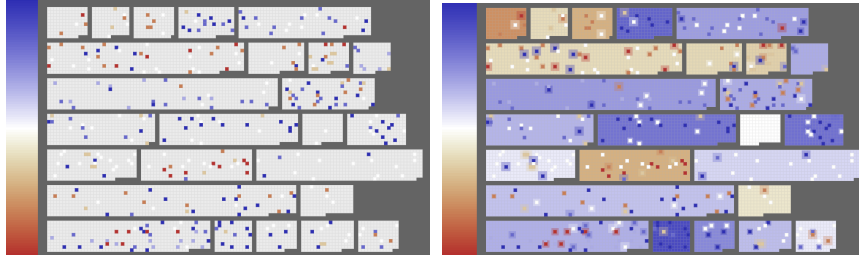


Figure 2.14: Pixel-based visualization, right image without visual boosting and left image shows visual boosting with background colouring and semi-transparent halos. Adopted from [154].

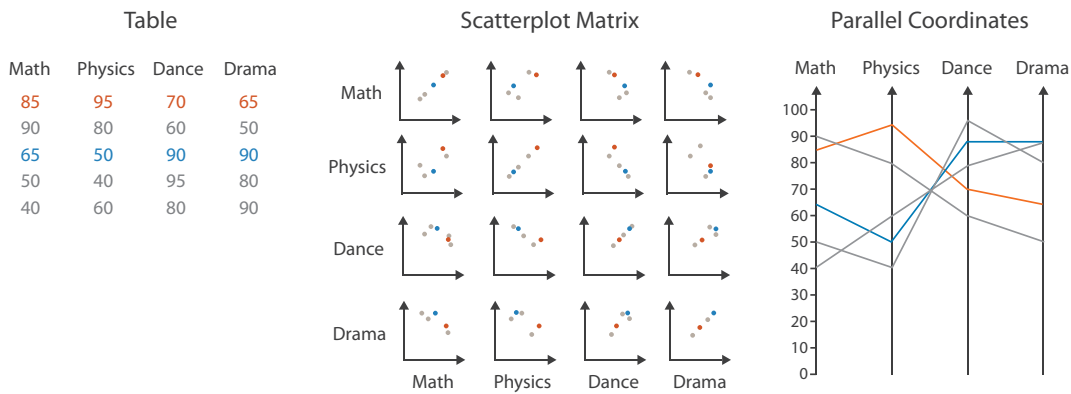


Figure 2.15: Multivariate information visualization techniques: tables (left), scatter plots (middle), and parallel coordinate plots (right). Adopted from [149].

are suitable for small number of data dimension. Using scatter plots, a N^2 plots are required to show data with dimension N . A review of parallel coordinates was presented in [102] and a study on perceiving patterns using parallel coordinates can be found in [110]. Parallel coordinates are difficult to learn and they take higher resolution. The fourth technique, glyph-based multivariate data visualization, is described in the next section.

2.3.2.1 Glyph-based Visualization

Glyphs are visual entities composed of several visual channels representing multivariate qualitative and/or quantitative attributes. Borgo *et al.* [28] described glyphs as small visual objects that have a meaning and involve learning. Ideally a glyph should be a compact and small visual object that can code multiple dimensions, as well as spatial information to convey meaning and it also requires learning. For example, Figure 2.16 (a) shows a simple glyph encoding four attributes of wind: direction, speed, temperature, and location. The use of metaphor makes the

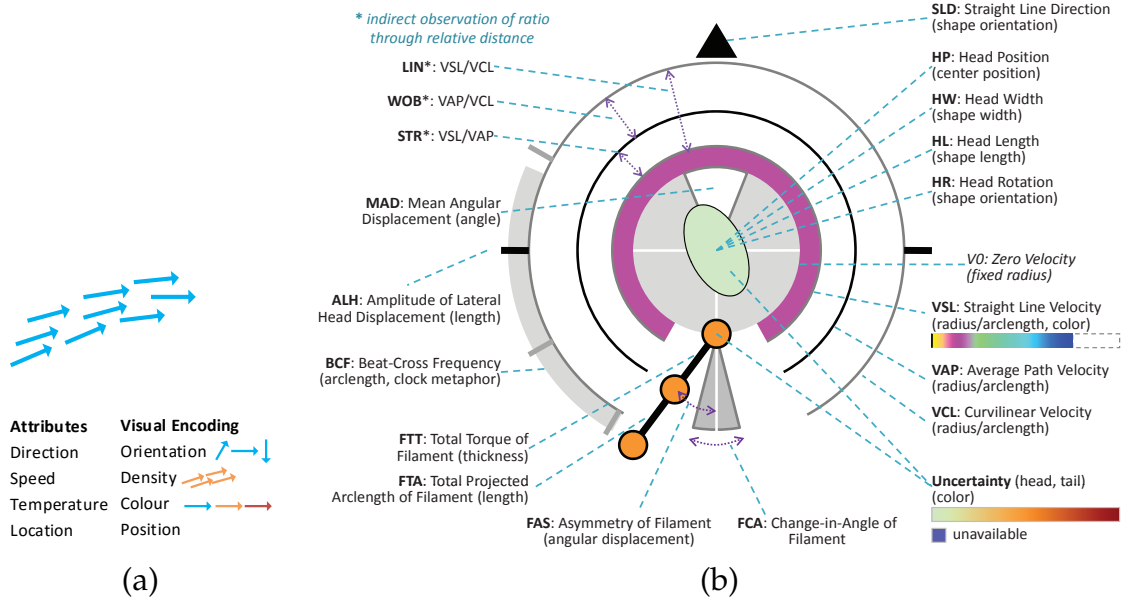


Figure 2.16: (a) Simple glyph depicting wind direction, speed, temperature, and location. (b) A complex glyph by Duffy *et al.* [72] encoding 23 attributes of sperm cells (adopted from [72]).

glyph intuitive to decode, e.g., colour for temperature, the orientation of an arrow showing direction, and position of the arrow for location. Duffy *et al.* [72] created a complex glyph that encodes 23 attributes of sperm, see Figure 2.16 (b). This glyph is used to summarise sperm video data in one second interval, thereby reducing the overall task of watching the videos repeatedly by the domain experts.

Glyphs design involves visual mapping and metaphoric abstraction. Therefore, the glyphs design process often goes through many iterations where data attributes are carefully assigned to visual channels, e.g., size, shape, colour, texture, orientation and so on. A well-designed glyph can facilitate effective information encoding, efficient learning and decoding whereas a bad glyph design can cause ambiguity and confusion [28]. Matt Ward [212] has written a survey on glyph placement strategies and presents a comprehensive taxonomy of different layout algorithms in 2D or 3D space. Maguire *et al.* [135] proposed a systematic guideline for glyph design by using a taxonomy. They have demonstrated the proposed approach by analysing and visualizing the content of a biological database. The perceptual strengths of different visual channels based on levels of organization can be found in Bertin's book [18]. Pre-attentive processing or the popout effect is studied extensively in the area of psychology and a summary of this work can be found in [135].

2.3.2.2 Application of Glyph-based Visualization

The glyphs were used to visualize diverse domain specific data. Glyph designs are largely influenced by particular application domains and tasks. Glyph designs could be generic such as using facial attributes (e.g., the length of the nose, the curvature of the mouth) to represent regularities and irregularities in data as presented by Chernoff [48] as early as 1973. Over the years more complex and application-specific glyphs have been proposed, see Figure 2.16 (b).

Matt Ward [212] classified different types of glyphs, as well as their placing strategies, and proposed rules for their usage and noted several bias issues in the context of information visualization. Borgo *et al.* presented numerous applications of glyphs to visualize different domain specific data in [28]. Here, we briefly summarise the glyph designs that have been proposed to visualize different domain specific data, e.g., search results, text and literature, computation, medical, geo-spatial and so on.

Search Results. Search results are often displayed as text in a list or in a tabular view, often showing 10-15 results per page. However, most of the information e.g., search space, search attributes, search history, search results, their distribution, clustering pattern and so on are not shown. Roberts *et al.* proposed glyphs to visualize Web search results [172]. Chau [42] proposed a flower metaphor based glyph to visualize more search attributes in a Web search results. Search keywords were shown using petals, outgoing links from a document were shown using leaves, the stem showed document length, and the supporting ground showed incoming links. Kachkaev *et al.* [113] proposed glyphs with parallel coordinate plots to display and explore survey results. Karve and Gleicher [116] presented a glyph-based visualization for search results from the Protein Data Bank.

Text and Literature. Glyphs are becoming a widely used technique for analytics and visualization of digital humanities domain creating large cultural datasets (e.g., history, text, literature, music, and so on). Visualization of document content and document relationships was described in [176]. Document contents are mapped to a shape which enables the identification of global content similarities and differences of multiple documents by comparing shapes. Closely related documents are clustered and blended as a single connected shape. There have been numerous efforts to visualize the scientific literature and citation network. Chaomei Chen [43] used network as well as cluster visualization techniques for visualizing trends and transient patterns in scientific literature. Abdul-Rahman *et al.* [1] pro-

posed a rule-based visualization technique for depicting the characteristics and positions of phonetic articulation in relation to the human vocal system. Their visualization system can display as many as 26 different poetic variables. This work was extended in [2] for visualizing the relationship and variance between different lines in a poem and in comparing different poems.

Computation. Environmental data has inherent uncertainty which is ignored in simple arrow glyph-based visualization. Therefore, a vector glyph was proposed in [221] to visualize uncertainty in winds and ocean currents. Botchen *et al.* [31] used glyphs to encode the position of objects, size, type of action, relation with other objects and uncertainty in the information, and created a summary from video surveillance systems. Legg *et al.* [128] used glyphs for annotation of rugby match videos. Such glyphs encode information about the type of event, the players, and the team involved. Chuah and Eick [53] presented glyphs to encode software code related information. The Bug glyph shows the number of lines added, the number of errors, the number of file changed, the languages used and the wheel glyph shows lines of code, the number of coders, errors, and changes in files.

Scientific Visualization. Glyphs have long been used in multivariate scientific visualization [87] among them flow visualization shows patterns in the flow of liquids and gases using glyphs [126]. Hlawatsch *et al.* [104] proposed a flow radar glyph to visualize flow directions by angles and time by radius in spherical coordinates. Such glyphs were used to visualize unsteady flow with static images, allowing for easier analysis of time-dependent phenomena compared to animated visualizations. Peng *et al.* [161] reduced visual clutter through composite glyphs. They showed the velocity and direction using a cluster of arrows.

Medical and Biology. Glyphs have widely been used in medical and biology data visualization. A detailed survey of the use of glyphs in medical visualization can be found in Ropinski's surveys [177–179]. Some examples of notable work in this area includes, Walton *et al.* [210] using glyphs to show multivariate attributes of Cardiovascular Magnetic Resonance (CMR) imaging data. An analysis and visualization of the content of a biological database was proposed in [134, 135, 145]. Duffy *et al.* [72] proposed a glyph-based visualization of 23 sperm attributes to summarize videos, see Figure 2.16 (b).

Geo-visualization. In geo-spatial visualization glyphs have long been used to overlay information on maps. A notable example is the one by John Snow who, in 1855, used glyphs to show cholera incidents. He used stacked rectangles to show

the number of incidences and the position in the map of London to show location. Sanyal *et al.* [183] used glyphs to show uncertainty in weather data. Ware and Plumlee [215] used glyphs for improved weather visualization.

2.3.3 Provenance Visualization

The quality or validity of data can be examined or reviewed in the context of its original source and transformations over time. The historical information that needs to be captured over time to support that process is called *Provenance Information*. A comprehensive review on different types of provenance information, use cases, and a framework for developing and evaluating a provenance application can be found in [124]. In this section we shall discuss provenance visualization techniques applied to a domain and/or domain specific data to support querying, workflow management, analytics and so on.

Web provenance data, e.g., search, navigation, click-stream, and query logs are extensively captured. Web search activities generate query logs [229], usability logs [92], and Web clickstreams [219]. Clickstreams [39, 127] support the analysis of Web traffic and the users' Web path navigation behaviour [51]. Such information is useful for understanding the effectiveness of Web marketing, advertising, and merchandising efforts. It also reveals how customers find stores, the products they browse, and the products they purchase and so on. Workflow visualization allows users to track provenance information such as evolution. Another notable application of provenance visualization is to explore the file system hierarchy and changes over the time. In this section, we shall present a survey of different provenance visualization techniques. Although provenance information is extensively captured in the Web search domain, there are no examples of work that have focused upon visualizing such provenance information to assist in search.

2.3.3.1 Text-view

Provenance management tools such as Haystack [107] used multiple dialogs or collapsible views to visualize documents, e-mail, appointments, tasks, and other information in a single window. The users' document and the metadata information are displayed in the main view, see Figure 2.17. Haystack uses a semi-structured data model to describe the relationship between the different documents and their metadata information. Stuff I've Seen (SIS) [74] was developed to provide a unified access to the provenance of what a user has seen on their desktop. SIS tool

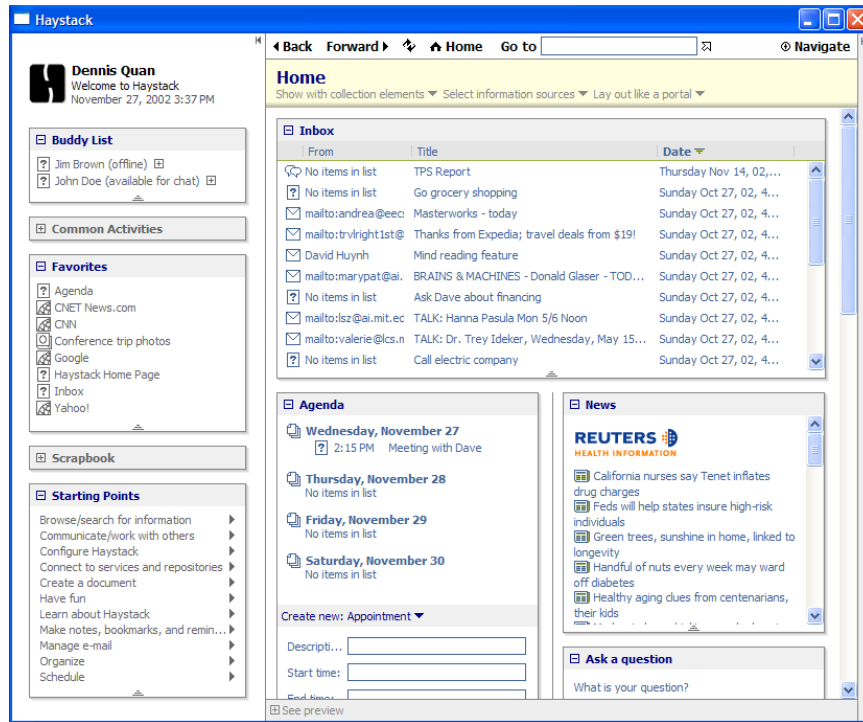


Figure 2.17: Example text-based provenance view, which shows different attributes and relationships between different documents. Adopted from [107].

uses multiple widgets to display, search, and filter desktop information using a text view, and it assists a user to tag their content with metadata. RAMP [160] automatically captures and displays workflow information in a text-based view.

2.3.3.2 Node-link Graph

The conventional visual encoding for provenance data is derived from the fields of network and graph visualization. The provenance visualization technique commonly used for provenance data is the node-link diagram [37] as shown in Figure 2.18. In node-link diagrams, the visual encoding of nodes are represented as dots or rectangles and edges or connections between nodes and are represented as lines or curves. WebQuilt [216] and Webviz [163] used node-link diagrams to visualize click-streams. Kreuseler *et al.* [124] used a tree structure to represent change histories and to calculate operations (e.g., undo and redo) of an exploratory data mining application. Behaviour Graphs [39] used state diagrams to visualize search structures on the Web for discovering usage patterns. The provenance information generated in a file systems operating systems, and large-scale distributed storage

2. BACKGROUND

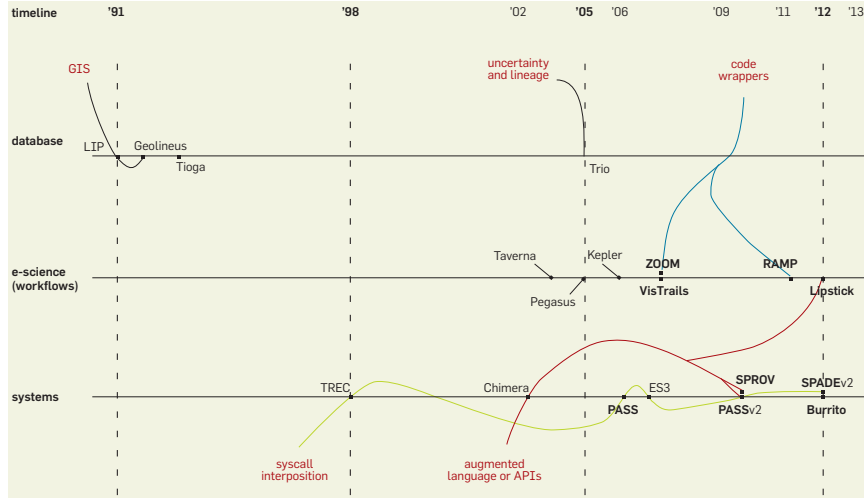


Figure 2.18: This example node-link visualization shows the work done in the area of operating systems, workflow, and database provenance visualization from 1991 to 2013. The nodes shown as dots and the links shown as lines connecting the nodes. Adopted from [37].

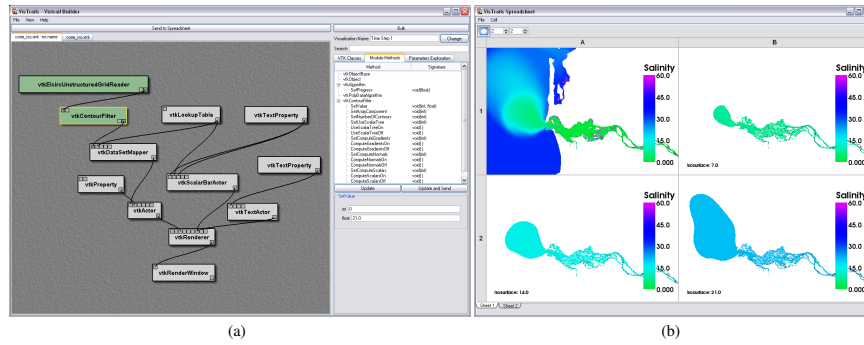


Figure 2.19: VisTrail: (a) node-link view of the dataflow, (b) spreadsheet view shows different visualizations of the data instances. Adopted from [36].

systems are visualized in [89,147,148] using the basic node-link graph or node-link based flowchart.

A minor improvement over these node-link diagrams featuring a variety of visual encoding methods, such as including directed edges [58], and collapsible summary nodes was proposed in Provenance Map Orbiter [133]. Wongsuphasawat and Gotz [223] used an improved node-link diagram for event sequence data from electronic medical records and sports events. GraphTrail [75] used node-link diagrams to visualize user interactions and historical information stored in a large network dataset. This tool also integrated charts, interaction, and a pivoting mechanisms with node-link diagram to support large network analysis. Chi [51] and Chen *et al.* [47] used forced-directed diagrams to visualize network data.

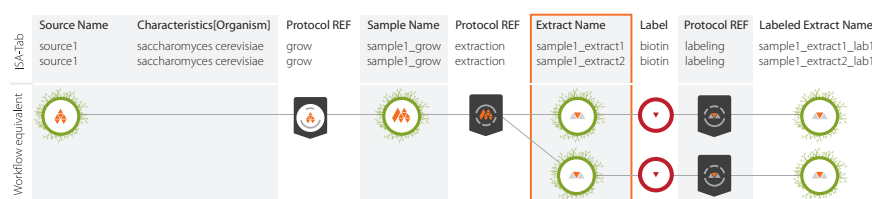


Figure 2.20: Workflow provenance information recorded in text using glyph-based graph visualization. Each node is enriched with multivariate glyphs to visualize more information. Adopted from [135].

The node-link based provenance view was further developed and enriched by adding an additional spreadsheet or thumbnail view for interactive and comparative visualization. For example, such diagrams were used for visualization of workflows in VisTrails [16, 36, 184, 193] (see Figure 2.19) and of scientific workflows in [22, 23, 57]. Walker *et al.* proposed a XML-based provenance model [143] to support data exploration and to visualize provenance in the context of Human Terrain Analysis in [209]. Probe-It [58] used a node-link and spreadsheet view to assist scientists to explore provenance data.

Amsterdamer *et al.* [6] proposed a workflow provenance visualization technique for Pig Latin (a query language) queries where different types of nodes were shown using a simple glyph. Furthermore, Maguire *et al.* [134, 135] improved the representation of the nodes in a node-link diagram by using glyphs to show multivariate information stored in each node and biological experiment workflows, see Figure 2.20.

2.3.3.3 Radial Layout-based Graph

It is difficult to scale a large volume of provenance datasets beyond a few hundred nodes using the node-link based graph. A radial-based layout can save space by placing the nodes radially. Borkin *et al.* [29] proposed the InProv tool to explore file system provenance information using the radial-based tree layout. The InProv also uses a time-based hierarchical node grouping method for file system provenance data. Such file system provenance visualization is particularly useful for understanding its complex hierarchical structure.

2.3.3.4 Other Provenance Visualization Techniques

Apart from widely used node-link and radial layout based provenance diagrams some works have used different visualization techniques, e.g., parallel coordinates,

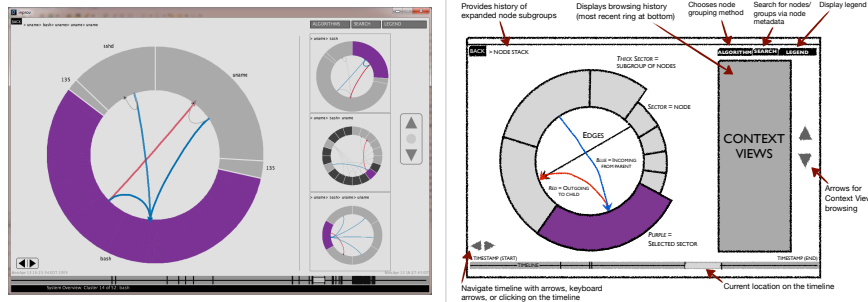


Figure 2.21: The InProv tool was developed to visualize file system provenance. The left image shows the interactions of the “bash” node with its parent nodes (connected by blue edges) and child nodes (connected by red edges). The right image shows “node stack” and “context views” to provide browsing history and location context. Adopted from [29].

line, and bar charts. Lee *et al.* [127] used parallel coordinates and star fields to visualize user paths and product performance. Lam *et al.* [101] provided users with multiple coordinated views for statistical as well as detailed analysis of Web search behaviors. The SDSS Log Viewer [229] was developed for identifying users' data-seeking behaviours from SQL query logs using time series and text visualization techniques. TrailExplorer2 [187] used stacked bars and pie charts to discover valuable information from large-scale Web click-streams. The Burrito [94] tool automated the experiment organization and note taking process to help researchers to focus on more substantive work. This tool allowed users to annotate the provenance notes (text) and make queries. Wei *et al.* [219] introduced an interactive clustering method to reveal user behaviour patterns from Web click-streams data. Whereas, Shi *et al.* [188] focused on analyzing the dynamics of user loyalty and defection using Web log entries.

2.4 Building Information Management and Search

Software companies particularly, CAD and BIM system development companies, have put a huge amount of effort to provide all the possible solutions required by the construction industry. In many cases, custom made solutions were commonly developed to meet the special requirements of the construction domain. For example, Asite²⁶ is an RDBMS-based solution specifically designed for construction document storing and retrieval. There have also been efforts to integrate information

26www.asite.com

retrieval mechanisms to CAD systems. For example, the literature (e.g., [88]) focuses on retrieval of BIM documents using the Industry Foundation Classes (IFC) data format. For that reason, we undertook a comprehensive survey on the CAD and BIM software and tools commonly used in construction industries. Table 2.1 lists such software and their functionalities.

It is relevant to note that none of the CAD systems, BIM software, and the construction document management systems support a large-scale document management and search cost-effectively.

Table 2.1: Key features of commercial BIM software (to continue in Table 2.2).

Company	Software	Model Creation (2D, 3D)	Model Navigation	Scheduling (Animation)	Cost Estimation	Simulation capability	Life-cycle	Maintenance	Collaboration	Other	Search Tasks
Autodesk	3Ds Max Design	✓	✓			✓		✓			
	Design Review									✓	
	Revit MEP	✓		✓		✓		✓			
	Ecotect Analysis					✓					
	Green Building Studio					✓	✓				
	Autodesk Impression									✓	
	Navisworks Manage		✓	✓		✓					
	Autodesk Quantity Takeoff				✓						
	Revit Architecture	✓				✓					
	Autodesk Revit Structure	✓									
	360 BIM Glue								✓		
	Robot Structural Analysis	✓				✓					
BEXL Consulting	Building Explorer		✓								
	Interactive Cost Estimating				✓						
	Intelligent Scheduling			✓							
IES	GAIA (Energy Calculation)									✓	
Aconex	Web-based BIM viewer								✓		
CADfaster	CADfaster Collaborate								✓		
Onuma Inc	Planning System				✓						
Solibri	Model Checker				✓						
	Model Viewer									✓	
Project Blueprint Ltd.	Zero Defects									✓	
Dassault Systemes	Live Buildings	✓									
Asite Solutions Ltd ²⁷	Asite										✓
Tekla	Structures	✓									
	BIMsight								✓		
	Cost Estimation				✓						

²⁷www.asite.com

2. BACKGROUND

Table 2.2: Continued from Table 2.1.

Company	Software	Model Creation (2D, 3D)	Model Navigation	Scheduling (Animation)	Cost Estimation	Simulation capability	Life-cycle	Maintenance	Collaboration	Other	Search Tasks
Bentley System	Architecture	✓									
	Structural									✓	
	Mechanical Building Systems									✓	
	Electrical System									✓	
	ConstructSim			✓							
	Navigator								✓		
	ProjectWise navigator			✓							
Innovaya	Visual BIM			✓							
	Visual Quantity Takeoff									✓	
	Visual Estimating				✓						
	Design Estimating									✓	
	Visual 4D Simulation			✓							
	Composers for Revit									✓	
	Composers for AutoCAD									✓	
Synchro Ltd	Synchro Professional			✓							
	Synchro Database Module									✓	
	Open Viewer									✓	
	Synchro Express									✓	
	Project Constructor									✓	
	Synchro Server									✓	
VICO Software	3D BIM for Visualization		✓								
	3D BIM for Clash Detection								✓		
	4D BIM for Scheduling and Production Control			✓							
	5D BIM for Estimating				✓						
	Construction Management Reporting								✓		
	6D BIM Models for FM & Maintenance							✓			
Nametschek	Graphisoft ArchiCAD	✓									
	Vectorworks	✓									
	Scia									✓	
Gehry Technologies	Digital Project:Designer	✓									
	Digital Project:Viewer			✓							
	Gteam								✓		
	Digital Project:Extensions									✓	

2.5 Summary

There have been numerous attempts to solve the document management and search problem by using databases as discussed in Section 2.2.1. Documented orientated and graph data models address the problem being faced by the relational data model, i.e., a future-proof complex schema. Resources are needed to transfer files

into any database. However, given the scale of files a large enterprise (e.g., Laing O'Rourke) has to deal with, the use of databases does not appear to be a feasible solution in terms of the resources that would be required to enter the files onto the database.

Desktop search tools and PIM systems described in Sections 2.2.2 and 2.2.3 do not scale well to a large-scale distributed file system search. As a result, the Information Retrieval community developed the concept of enterprise search engine to address the large-scale distributed file system search problem.

However, searching through a large-scale data infrastructure using an enterprise search engine as discussed in Section 2.2.4 has certain limitations, these relate to and are caused by: (a) search queries often returning a long list of file displayed as text, and (b) the search results often include false positives and false negatives. Although query expansion techniques (e.g., [19, 97, 151]) can moderately improve the recall, i.e, false negatives, it does not nevertheless improve the precision.

The precision can be improved by introducing improved ranking and active learning mechanisms. The active learning has long been tried in the Web search domain, however it is yet to be applied to enterprise search [54]. In contrast, the active learning mechanisms applied to Web search cannot be directly applied to enterprise search, this is due to the fact that the structure of enterprise data is different from the Web data. For example, ranking techniques (e.g., PageRank [157] for anchor texts) are not effective in enterprises search [100]. The ontology based techniques described in Section 2.2.4.1 (e.g., [235]) cannot be applied directly to file search.

Thus no search can guarantee to find all the relevant file or how to correctly specify the search criteria. Therefore, it is essential to improve the search time with visualization capabilities, e.g., assisting users to identify false positives and false negatives, improve search collaboration, and assist in formulation/reformulation. Visualization has yet to be seriously considered to assist in enterprise search. The Lucene-based²⁸ enterprise search engine, Elasticsearch²⁹ integrates Kibana³⁰ for basic enterprise search engine data visualization using area chart, line chart, pie chart, bar chart, and tile map.

Treemaps [111, 190] and Cushion treemaps [208] are commonly used space efficient visualization tools for file systems. However, treemaps do not support docu-

²⁸lucene.apache.org

²⁹www.elastic.co

³⁰www.elastic.co/guide/en/kibana/current/tutorial-visualizing.html, (date of access: 09 Sept 2015)

ment search. Moreover, they cannot display multivariate file system information. Therefore how enterprise search results are displayed and their distribution in the search space has not yet been addressed.

Search provenance and its visualization may provide an effective means for users to formulate/reformulate queries, facilitating external memorization in a search task that may be performed over a period of time, and by multiple users collaboratively. Search provenance information can also help administrators to learn from their historical searches, mistakes and good practices. The provenance visualization techniques are discussed in Section 2.3.3. Among all the provenance visualization techniques only a few techniques have been applied to search (e.g., [39, 51, 92, 127, 219, 229]), with the majority being used and focused upon improving Web marketing, advertising, and merchandising. Therefore, it is essential to develop search provenance visualization techniques to assist users in search.

3

System Overview

Contents

3.1	Introduction	41
3.2	Enterprise Data Management and Search	42
3.3	System Overview	47
3.4	Business Cost Estimation	57
3.5	Summary	61

3.1 Introduction

Large enterprises typically have diverse functional areas where different varieties of data are generated throughout the life-cycle of a project. Throughout this life-cycle varieties of files are generated that are owned by numerous and different stakeholders. The files are, in the first place, stored in various storage systems, e.g., a distributed file system and databases. Multiple stakeholders are interested in different files. This posed a significant challenge for Laing O'Rourke, i.e., how could you make all the information quickly searchable at the end of each building construction project?

We have been working collaboratively with our industrial partner, Laing O'Rourke. We have had opportunities to familiarize ourselves with their existing document management systems, the distributed file system, and repositories used by different groups across their organization. We identified their requirements, the use cases, and the technical challenges of the system. The construction industry has

a large number of stakeholders (e.g., clients, architects, suppliers, local governments, transport authorities) and many teams within each construction enterprise. Therefore the need to quickly locate relevant files in this ocean of files is essential. Navigating through directories, and ever expanding and collapsing hierarchies of folders is not an efficient way to access stored documents.

3.2 Enterprise Data Management and Search

In order to develop a cost effective solution to support search over all the files stored in the distributed file system of an enterprise, we considered three main strategies, namely: (1) *A Database*, (2) *An Enterprise Search Engine*, and (3) *A Visualization-assisted Enterprise Search Engine*. As part of the overall evaluation of these three strategies, Figure 3.1, this research project was formulated to study the feasibility of these strategies.

3.2.1 Database

A traditional method of managing data is to use a database management system. This strategy requires each building project to maintain a high-quality database of all documents, from which a collection of the required files can easily be found upon the conclusion of a project. However, not every data management and analysis problem can be best solved using a database. Although we can classify data, trawling through hundreds or thousands of categories and subcategories of data is no longer an efficient method for searching for information. Over time, the amount of data available has become so vast that an alternate, more dynamic way of finding information is needed.

The main challenges of the database-based approach are: (i) the commitments of data archivists for each project, and the risk of a single-point of failure within each project, (ii) developing a database schema that is scalable to future changes, and finally, and most importantly, (iii) a significant cost and expense is involved in entering the high volume of files to a database.

In database strategy, the primary tasks may include:

- Each construction project should have one or more administrators employed locally. An administrator regularly captures and stores the files generated by the project through in a distributed file system.

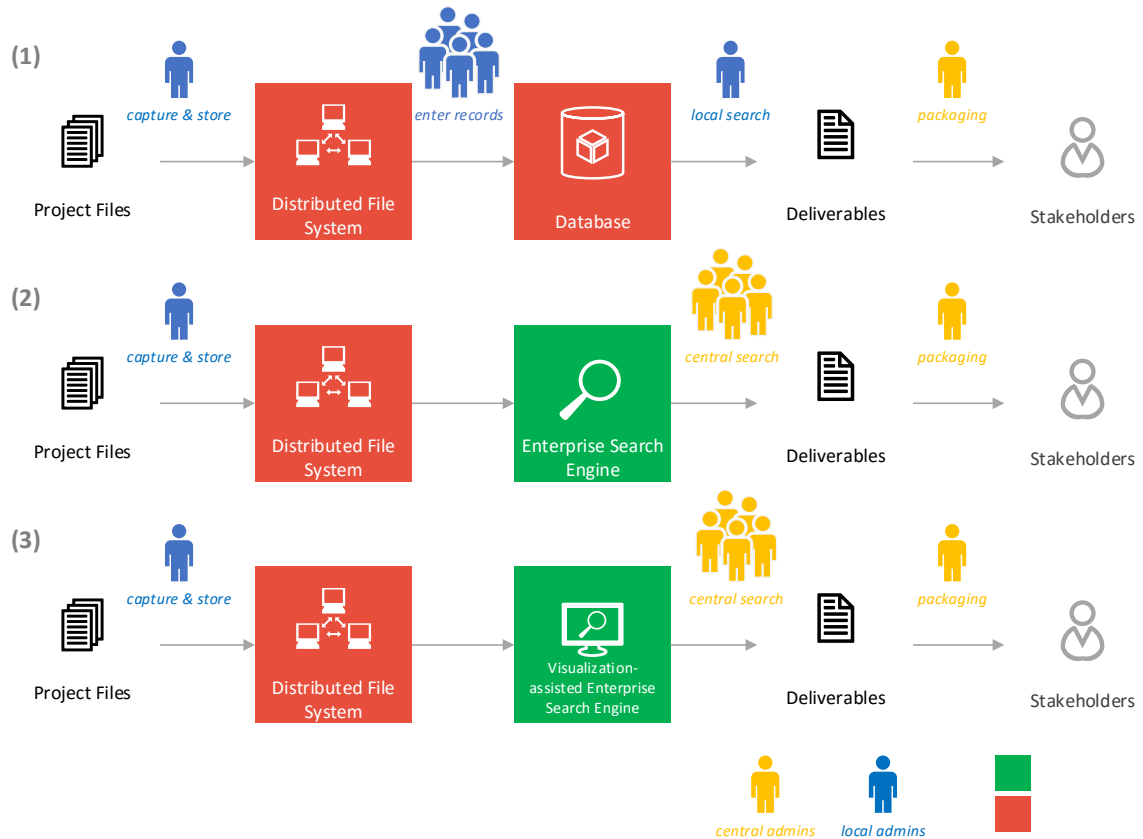


Figure 3.1: The three different strategies considered are illustrated in this figure. (1) In *Database* the task may include capture and storage in a file system, to enter files into a database, search in the database, and finally package and delivery. (2) In *Enterprise Search Engine* the task of recording each file in a database is eliminated and here the file search over a distributed file system is through an enterprise search engine. (3) In *Visualization-assisted Enterprise Search Engine* the search using the enterprise search engine is supported by visualization capabilities.

- An administrator enter each file details to a database to make them quickly and accurately searchable at the end of the construction project or whenever needed.
- At the end of the construction project, a local administrator can search in the database for the files relevant to the customer of that particular project.
- Finally, the organization should have a central team of administrators. The project relevant files are quality checked, packaged, and delivered to the customer by the central administrators.

3.2.2 Enterprise Search Engine

In an enterprise search engine strategy the primary tasks may include:

- Each construction project should have one or more administrators employed locally. An administrator regularly captures and stores the files generated by the project in a distributed file system.
- At the end of the project the central administrators can search for the files that are stored in the distributed file system with all the relevant information or file attributes.
- Finally, the project relevant files are quality checked, packaged, and delivered to the customer by the central administrators.

Unlike a database-based search, the search results in an enterprise search engine are not consistent and it has some major problems.

Problem I: Multivariate Information. The first challenge of the enterprise search engine is to deal with multivariate search results, which are typically presented in a list or tabular view accompanied by a sorting mechanism. Viewing a textual list of search results has a major shortcoming. It is difficult for a viewer to build a mental model about where these files are, and how they related to each other. Although we can sort files based on a file attribute, it is difficult to sort multiple variables. For example, when we are looking for files meeting more than one criteria it is difficult to sort them based on multiple criteria (e.g., file types, size, date, time, user, etc.).

Problem II: Search Quality. The second major challenge of the enterprise search engine is to deal with false positives and false negatives during the search process. Given a set of search criteria, a search engine executes a query, returning a set of results. Among these results, there are: *True Positives* – The files which have met the search criteria, and are also among those required by the user. *False Positives* – The files which have met the search criteria, but are not among those required by the user.

Meanwhile, among the vast number of files not returned by the search engine, there are: *True Negatives* – The files which have not met the search criteria, and are not among those required by the user. *False Negative* – The files which have not met the search criteria, but are actually among those required by the user.

Traditional representation of search results, a list or tabular view, does not show the search context. For example, the search context may include the information

about search space such as file system structure, number of files, location of the files, and so on.

Without such contextual information it is difficult for a viewer to build a mental model about whether there are false negatives and what search criteria may reveal them.

Problem III: Search Collaboration. The third challenge of the enterprise search engine is that it does not support search collaboration. In an industrial-scale search scenario where thousands of files to be found, a search task may continue for many days and may require collaborative team effort. Using a conventional enterprise search engine without provenance capture, retrieval, and visualization, collaborator will encounter difficulties.

For example, in a collaborative search scenario, an administrator (a) may search for few days. In the following week few more administrators (b and c) may join or take over. At the end another administrator (d) may do final quality check. When the search engine returns hundreds or thousands of files, it is difficult for the administrators to judge what may have been seen before. It is difficult for the administrators to pick up search activities after a weekend or from another colleague.

3.2.3 Visualization-assisted Enterprise Search Engine

The challenges with enterprise search engines have lead us to propose improvements by utilizing both machine and human capabilities, thereby assisting enterprise search with visualization capabilities. As part of the feasibility study we have developed an enterprise search engine and a search interface assisted by visualization capabilities, enabling the experimentation with visualization facilities that could not be done with a general purpose search system.

3.2.3.1 Why Visualization?

I. Multivariate Information. There are four main techniques to visualize multivariate information (as described in Section 2.3.2). Tables and scatter plots are suitable for small number of data dimensions. As parallel coordinates are difficult to learn and take much higher resolution, therefore we considered glyphs to be a desirable option. The glyph representation of the individual file attributes can enable users to quickly identify the relevant files. For example, given a list of search

results a user can quickly identify the files based on a combination of file attributes (e.g., file types, size, date, or time) displayed using glyphs.

II. Search Context. In an industrial-scale search, contextual information about the search space can help administrators to identify and pick out *true positives* quickly among *false positives*. For example, displaying the distribution of the search results in relation to the file system layout can intuitively convey the information about the search quality. This can enable administrators to identify *false positives* based on spatial awareness as well as the glyph representations of the files returned by the search.

In addition, the visual representation can also stimulate hypotheses of *false negatives*. For example, an administrator may expect certain files from a certain region. When a set of search criteria does not reveal those files, this indicates the presence of potential *false negatives*. Similarly, when the administrator notices certain files from a region that is related to a different project and should not be present in the search results these files are considered to be potential *false positives*.

The combined focus (glyph representation of files) and context (visual representation of the files system) can help user to understand (a) the overall context of the search result ('where in the information space is the result?') and (b) multivariate attributes of each search result ('what are the file types? how accurate is this result?').

III. Search Provenance. In order to enable such tasks to be performed efficiently and effectively, it is necessary for team members to access and share the search provenance information. Search provenance information may include the historical and ongoing activities related to a search task. Traditionally provenance informations are captured in a text. When a search process continues for days, it is difficult to interpret such a large number of provenance records using a text-based representation.

In addition, in an industrial scale search setting when multiple administrators collaborate together, the provenance records may become more complex. For example, an administrator may be interested in knowing his/her previous search criteria, or the overlap with other administrators. For such complex provenance records, visualization can provide an enabling tool for users to observe provenance information at a glance [36], improving the cost-effectiveness in *collaborative search tasks*, especially in an industrial setting. Such provenance information can potentially be used to identify and highlight false positives, i.e., the irrelevant files that are appearing in search criteria repeatedly.

For an observational task [46] such as file search, visualization can help a viewer obtain an overview quickly, and can provide a viewer with external memorization compensating for the rather limited working memory of a human. By reducing the time and cognitive load required for building a mental model and memorization, a viewer can devote more time and cognitive load to thinking (e.g., reasoning false positives and hypothesizing false negatives) [45].

3.3 System Overview

In order to provide immediate responses to the search queries, it is necessary to crawl the distributed files system and to maintain an index of the files which can be searched quickly. In order to support query enrichment based on semantics and ranking of the search results ontologies are needed. Although ontologies can be improved manually, automatic learning of the ontologies using the users' relevance judgement information is essential. In order to provide an interactive visual search, and an identification of false positives and false negatives we need a focus+context view of the search results and search space. The search space information can also be stored in the index. In order to visualize false positives and to support search collaboration we needed a provenance management and provenance visualization facility. The ontologies can support the provenance management facility.

In this work, we have designed and prototyped a visualization-assisted enterprise search engine (VAESE). The VAESE has an enterprise search engine running as a server and a Web-based *Search Interface*. The enterprise search engine consists of a collection of functional modules, which reside between the *Search Interface* and the distributed file system. These modules provide a real-time service to the *Search Interface*, see Figure 3.2. The *Search Interface* is integrated with multiple visualization facilities (see Figure 3.3) to assist users during search. The visualization capabilities are closely integrated with the search engine components. We, hereby, give an overview of a search process using a flowchart, see Figure 3.4. In this flowchart we only describe the main modules and the key components of each module. We also describe the interfaces handling the information exchanges between the enterprise search engine and the *Search Interface* in JSON data format.

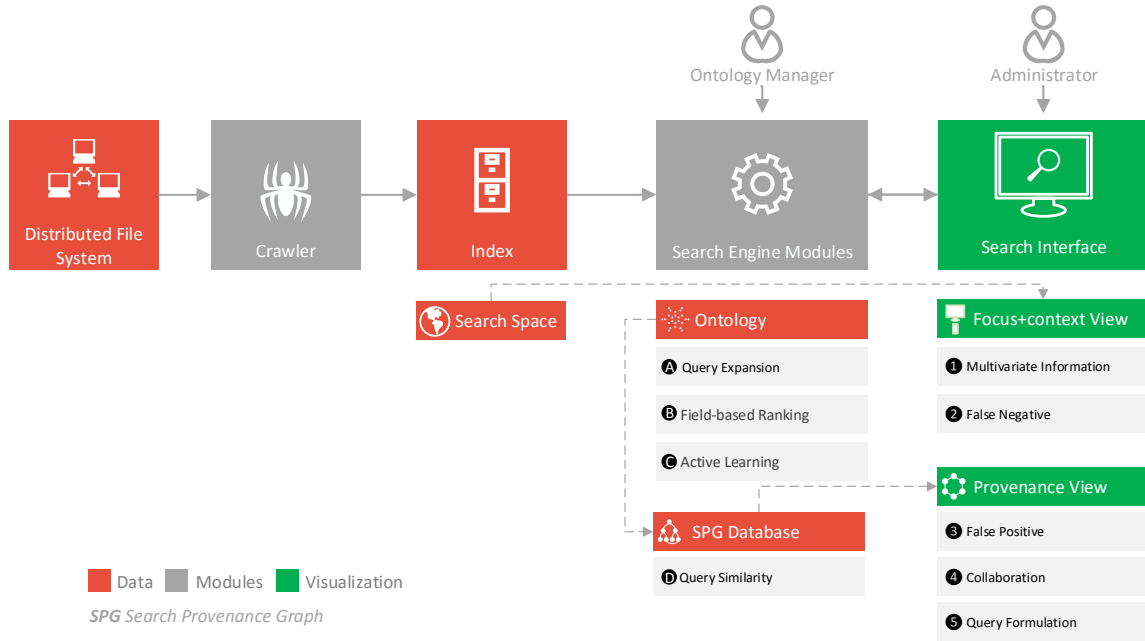


Figure 3.2: The modules of the enterprise search engine.

The main visual and interactive features of the interface were implemented using AngularJS¹ and the D3.js library [30]. The search engine modules were implemented in C++. The interface between the search engine and the Web application is implemented in Node.js².

3.3.1 Search Query Form

User initiates a search process by formulating a search criteria using a **Search Query Form**, see Figure 3.4 (A). A search query is composed of one or a few keywords and/or some constraints of file attributes, such as type, size and date. For example, the search criteria formulated here is to look for files related to “building material” of type “CAD”, “Document”, and “Spreadsheet”, size between 1KB to 200MB, and created during 7th June 2012 to 19th December 2014.

Once the user formulates a search criteria and submits it, the `SearchQuery` interface makes a http request to the search engine. The http request is served by the **Query Processing** module of the search engine.

¹angularjs.org

²nodejs.org

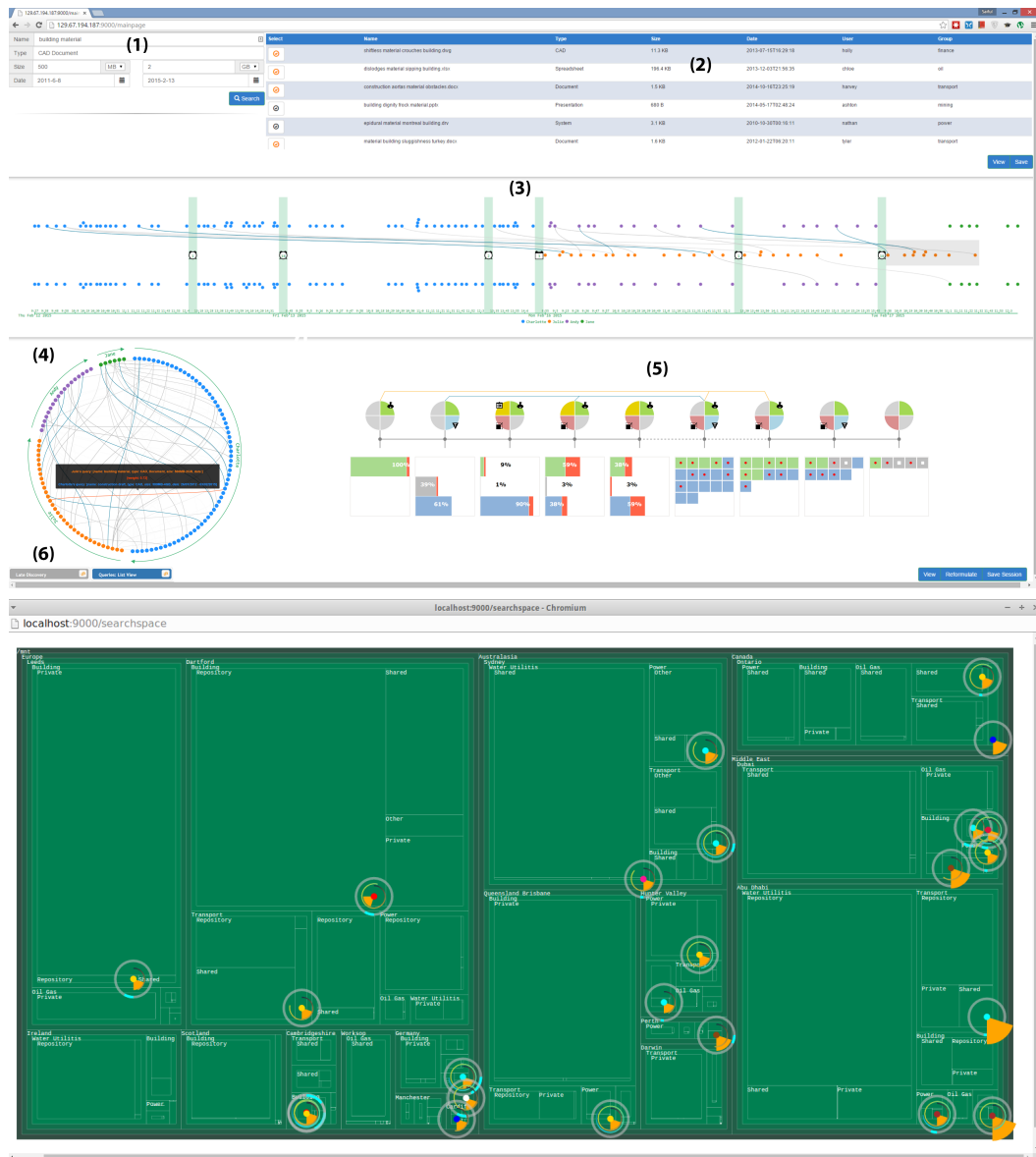


Figure 3.3: Top window: an example screenshot of the search interface, which is designed to be used with two computer displays. The components shown in the first display are (1) the search query form, (2) the tabular view of search results, (3) the graph view of SPG illustrating how the user (focus) collaborated with others, (4) the collaboration summary view, illustrates the similarity between the search queries performed by the administrators, (5) glyph-based view, uses multivariate glyphs for search queries and search results which are shown above and below the central line respectively, and (6) are the dialogs for showing the queries and the late-discovered files. Bottom window: the component shown in the second display is a treemap visualization of a file system and the search results encoded in multivariate glyphs.

3. SYSTEM OVERVIEW

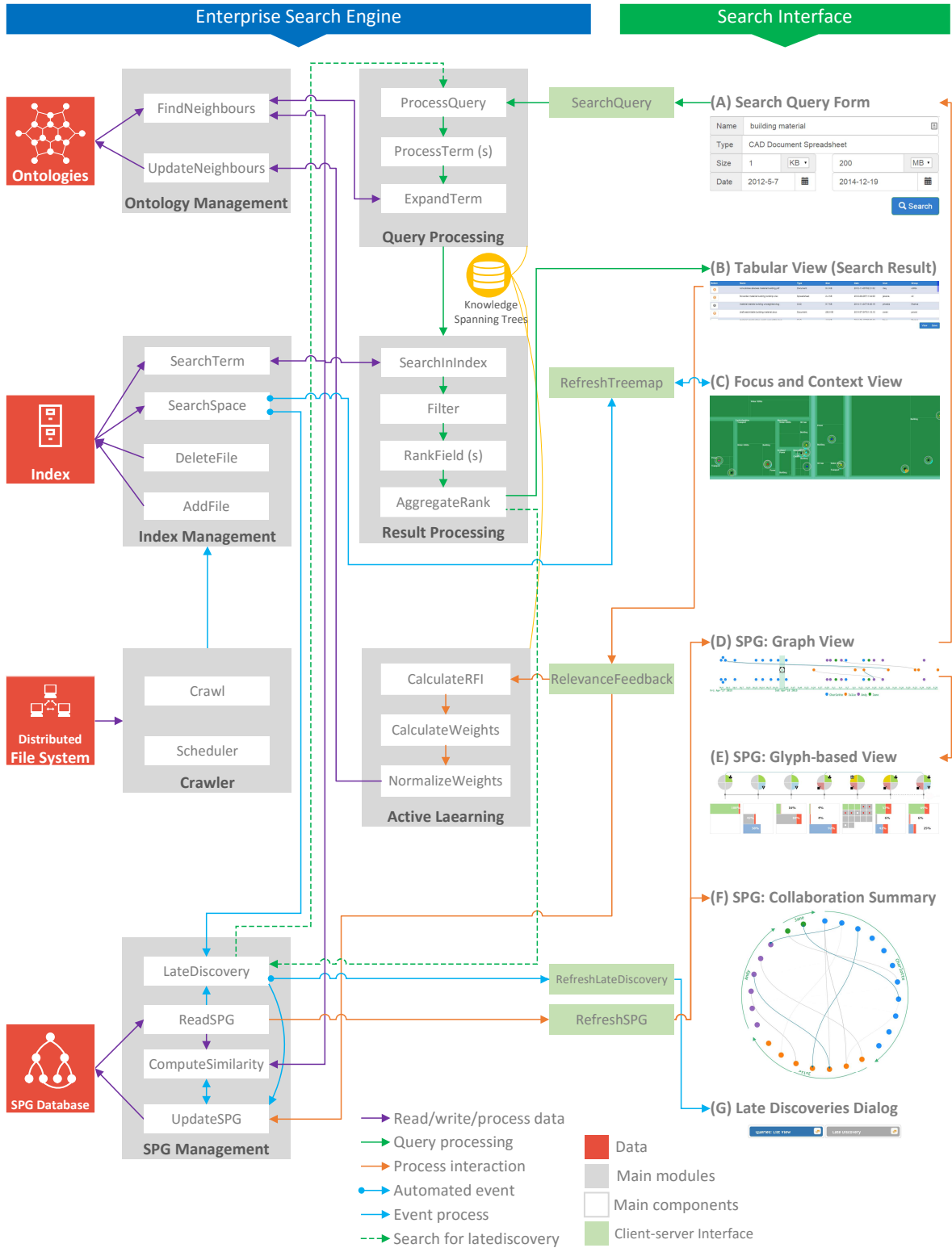


Figure 3.4: A flowchart of the VAESE system illustrating a typical search process and the visualization of the search results and provenance.

3.3.2 Query Processing

This module improves a seed query to increase the quality of the retrieval of the search results. The files and folders created during each project may have a different name convention, errors in file naming and placement are also expected. Therefore, improving the search criteria in order to be able to retrieve all the relevant documents is necessary. This includes, fixing spelling errors in user-entered terms and stemming (e.g., “building” to “build”) each term in the search query to match files named with various morphological forms. It is also desirable to derive a set of extended search criteria, e.g., closely-related terms and similar file types to widen the search space. For example, for the search term “building”, it is often desirable to search for “construction”, “manufacture”, etc. and the file type “CAD” should include file types, e.g., “3dm”, “3ds”, “mdx” and so on.

The **Query Processing** module takes a search query as an input and gives an expanded search criteria as an output. At first, the `ProcessQuery` component parses the query and derives a set of search criteria: keywords or terms and different field constraints to be used as a filtration conditions. For each derived term from the original query, the `ProcessTerm` component invokes the `ExpandTerm` component to enrich the search criteria. The `ExpandTerm` component invokes the **Ontology Management** module to search for similar terms or terms with other morphological forms. For each original term the **Ontology Management** module returns a knowledge spanning tree containing the extended terms in a hierarchical structure. The `ProcessTerm` component manages the knowledge spanning tree derived for each original term, which is later used by other modules, e.g., during ranking and active learning. The detailed description of the query expansion technique can be found in Section 4.4.2. Finally, the expanded search criteria is passed to the **Result Processing** module.

3.3.3 Result Processing

Search results often consist of a long and lengthy list of files, it is therefore necessary to eliminate the files based on any constraints defined in the search criteria. It is also necessary to assign a score to each file based on their expected relevancy to assist in this process.

The **Result Processing** module takes an expanded search criteria as an input and returns to the *Search Interface* a list of relevant results ranked according to their relevance. Initially, for each of the expanded search terms stored in the knowledge

spanning trees, the `SearchInIndex` component searches for the relevant files using the **Index Management** module. Once all the relevant files are retrieved they are passed to the `Filter` component. The `Filter` component filters the results based on the constraints defined in the search criteria. Thereafter, the filtered results are scored for each file attribute (e.g., name or keyword, type, size, date) using the knowledge stored in the corresponding **Ontologies** and the knowledge spanning trees. Finally, the `AggregateRank` component computes the final relevance score combining the scores for each field. The detailed description of the ranking process can be found in Section 4.4.3. The search results along with their relevance rank are returned to the *Search Interface*.

3.3.4 Search Result Visualization

The *Search Interface* first displays the results in a **Tabular View** sorted in descending order according to their relevance rank, see Figure 3.4 (B).

However, the downsides of a list or tabular view is the lack of contextual information. Therefore, we have developed a **Treemap View** to visualize the search results, this can offer a cost-effective means of showing the contextual information for a set of search results.

In this visualization, the *Search Space* (context) is visualized using a treemap and the search results (focus) are overlaid on top of the treemap. When the *Search Interface* loads in the Web-browser, the `RefreshTreemap` interface fetches the *Search Space* information stored in the **Index**. Each search result is encoded as a multivariate file system glyph and overlaid on top of the treemap, see Figure 3.4 (C). If there are already exiting glyphs from the previous search they are flushed away and new glyphs are drawn. This visualization can support quick *Visual Search*, identification of *false positives*, and *false negatives*. The details of this visualization work can be found in Chapter 5.

The user can scroll or navigate through the results displayed in the **Tabular View** as well as in the **Treemap View**. The user can select the relevant files, review the list of selected files, and finally save the relevant files as a portfolio. When the user saves the selected files as a portfolio, the `RelavanceFeedback` interface sends this information (e.g., the query, the search results, and the selected file information) to the search engine. This information is processed by the **Active Learning** module of the search engine. The same information is also stored in a **SPG Database**.

3.3.5 Ontology Management

Ontologies are used to provide the knowledge-based support of the search engine modules. The system is equipped with multiple ontologies, including a *Term Ontology*, a *Type Ontology*, a *Size Ontology*, and a *Date-time Ontology*. For example, the *Type Ontology* stores the basic terms in English (including abbreviations) and the weighted relationship between them. The *Type Ontology* captures the relationship between categories of file types and terms in English. The details of the ontologies are presented in Section 4.4.1.

Ontologies are used by the **Query Processing** module to expand a set of search criteria specified by a user. Given one or a number of input terms in the original query, the `FindNeighbours` component first locates the corresponding source nodes in the *Term Ontology*. Starting from each source node, it activates a recursive sequence for determining a *Knowledge Spanning Tree*, connecting all nodes with an appropriate weight, and reaching out to all its destination nodes. The ontologies are automatically improved through an **Active Learning** module. After each active learning process, the `UpdateNeighbours` component receives the updated weights and writes it back in the ontologies.

The ontologies can also be manually improved by a relevance engineer or ontology manager who is knowledgeable on the enterprise domain and data using a graphical user interface. The relevance engineer would identify new concepts and the relationships that are relevant to the search and update them in the ontologies.

3.3.6 Active Learning

However, manually tuning the weights of the **Ontologies** is a very tedious, time consuming and thus an expensive task. Therefore, the search engine implements a semi-supervised learning algorithm able to actively utilize the user interactions, i.e., the users' actions of selecting files to improve the weights of the connections in the **Ontologies**. Thereby, the search engine learns to improve iteratively by utilizing the user's relevance judgement information.

The **Active Learning** module processes relevance feedback information (i.e., true positives) which serve for the automated improvement of the ontologies. At first, `CalculateRFI` computes a Relevance Feedback Index (RFI), see Section 4.4.4. Thereafter, the `CalculateWeights` component increases or decreases the weights along the connections of the ontologies based on the RFI values, see Section 4.4.5. Finally, the `NormalizeWeights` component normalizes the weights

between $[0, 1]$ by preserving the average weight equal to 0.5, see Section 4.4.6. The **Ontology Management** module updates the computed weights in the **Ontologies**.

3.3.7 Index Management

In a large engineering enterprise, a significant volume of files are generated, this can potentially be in the scale of billions, and these are typically stored in a file system distributed across many offices located in different regions across the globe. Due to the size and the distributed nature of the file system, scanning the entire file system in real-time at an interactive speed is not feasible. The search engine gathers the information about the files and folders of the **Distributed File System** during the **Crawl** process and then creates an **Index**. Much like the index in the back of a book, the search engine index includes information about files and folders names (keywords) and their locations. During search, the search engine looks up search terms in the **Index** to find the information regarding the relevant files at an interactive speed. Nevertheless, to optimize the scalability of the **Index**, if required, it can only store the information about the files and folders that have been and may potentially be related to the recent search queries.

The **Index Management** module supports the **Result Processing** module to discover search results in real-time. The `SearchTerm` and `SearchSpace` components assist in the search process, and the `AddFile` and `DeleteFile` components assist in **Index** maintenance. Based on the information received from the **Crawler**, the `AddFile` and `DeleteFile` components add or delete entries in the **Index**. The `AddFile` component collects, scans, and parses the metadata of the files fetched by the **Crawler** and creates a data structure which can be searched quickly. At first, it analyzes each parsed word to remove stop words, or to normalize the words. This stage optimizes the parsed result. Following this it creates a data structure of inverted indexes. Details of the data structures and the algorithms used to create and update the **Index** are described in Appendix A.

Given a term, the `SearchTerm` component looks for the relevant files in the **Index**. The entire *Search Space* information is also computed based on the entries present in the **Index**, which is used by the **Treemap View** to draw the search space. Whenever, the **Index** is updated the `SearchSpace` component generates an event notifying the Web browser for the updating of the treemap. The same notification event is also sent to the **SPG Management** module.

3.3.8 Crawler

A **Crawler** is a set of agents that scan through the distributed file system in an autonomous manner and at predefined scheduled intervals. It fetches the information related to files and folders of the **Distributed File System** for later processing by a search engine, that will **Index** the information to provide fast searches. Moreover, in an enterprise, the update, deletion, or creation of new files occurs very frequently, therefore, the **Crawler** keeps the **Index** up-to-date.

The `Scheduler` component performs a time-based job scheduling of the `Crawl` component based on a configuration file and the GNU/Linux utility “Cron”. The `Crawl` component fetches all the information such as when it discovers a new file, or a modification/deletion of an already indexed file. In instances of a very large search space, the crawled information can be limited to queries potentially related to future search (similar to anticipatory fetch). This component determines if an existing entry has to be deleted or a new entry has to be added and, accordingly, invokes the maintenance component of the **Index Management** module which takes care of the actual **Index** update task.

3.3.9 SPG Management

Provenance information can assist users to explore search criteria and to reformulate search queries progressively by observing and learning from successes and errors in their earlier search results. Moreover, in an enterprise, the users’ goal is to work efficiently in collaboration with others to find all the required files for a project using the fewest possible number of queries. When a search process takes a few days and/or involves more than one user, visualization support for search provenance is useful.

The **SPG Management** module processes and maintains the SPG (Search Provenance Graph) related information (e.g., the queries performed, the search results received, and the selected file information) in a **SPG Database**. It also provides provenance data to the provenance visualization facility. The details of the SPG can be found in Chapter 6.

When the `UpdateSPG` component receives the relevance feedback information from the *User Interface*, it first fetches all the related queries performed earlier by the `ReadSPG` component. Thereafter, the `ComputeSimilarity` component uses the semantic information stored in the *Term Ontology* to compute the similarity

between the current query and the past queries. Finally, the information is updated in the **SPG Database**.

Whenever this module receives an **Index** update event from the **Index Management** module, it looks for and notifies the user about newly discovered files related to their previous searches. In order to find the newly discovered files, it searches for all the queries related to the ongoing project performed earlier with the new **Index**. It then notifies the Web-browser through the `RefreshLateDiscovery` interface.

3.3.10 Search Provenance Visualization

The SPG information received from the **SPG Management** module is processed by the `RefreshSPG` interface to visualize it using three provenance views: a **Graph View**, a **Glyph-based View**, and a **Collaboration Summary View**.

The **Graph View** shows a subgraph of the SPG in a graph view, see Figure 3.4 (D). Each search query is mapped to a node (dot) of the graph and shown as a dot with different colour coding for different users. Each dot is positioned primarily according to the time when the search event took place. In the central line, we show the focus nodes, i.e, the user performing the search and the other nodes (users working on the same project) are shown above as well as below the central line. A link shows the relationship between two nodes.

The **Glyph-based View** shows a set of queries selected in the **Graph View** using multivariate glyphs, see Figure 3.4 (E). This view shows some of the basic information needed that may aid in the assessment of the effectiveness of a query, and assist in better query reformulation. Such information includes (i) how many files were returned by the *current* query concerned, (ii) how many are *common* with the previous query, (iii) how many files returned by the *previous* query are no longer in this query, (iv) how many selected files (i.e., true positives) are amongst each of the above categories of files, and (v) how many files have been returned many times but never selected (likely to be false positives).

The **Collaboration Summary View** visualizes the degree of similarity between the searches performed by the users working collaboratively, see Figure 3.4 (F). It shows how their queries relate to each other, the explicit and implicit connections are visualized using different colour coding. Explicit connections indicate that the users formulated queries are based on each other, i.e., collaborated well with each other. On the other hand, the degree of implicit connections (system computed

similarity) will indicate repeated searches and thus suggesting that collaboration can be improved.

When the `RefreshLateDiscovery` interface receives any information about newly discovered files, it creates an entry in the **Late Discoveries Dialog** in the *Search Interface*, see Figure 3.4 (G). At this point, if required, the user can perform the queries again to retrieve those newly discovered files. A detailed description of SPG and the provenance visualizations can be found in Chapter 6.

3.4 Business Cost Estimation

In this section we attempt to estimate the average cost involved in performing differing and relevant tasks using the the database, enterprise search engine, and VAESE. The cost estimation was undertaken with the help of experts from Laing O'Rourke and it is based on statistical summary data of several previous and/or ongoing projects and the experts day-to-day experience in handling file management and search.

The cost estimation presented here is a typical business cost estimation which is done by taking averages over a period of time, e.g., the average of how many files were generated, how many files were stored in the distributed file system, how many files were entered in the database, how many files were searched and delivered to the customer, and so on. It is not a measurement of a *single task* or a *button pressing* exercise where a task is performed by pressing a button to indicate users response. For example, if a file can be entered or stored by simply pressing a button it may only take 15 seconds or a keyword-based database entry for 15-20 keywords by simply filling a form which may take 3-5 minutes or less.

3.4.1 Project Value

To help us estimate the cost, our industrial partner, Laing O'Rourke, provided us with statistics with regards to several example projects. These statistics regarding three example projects are presented in Table 3.1.

- *Education Building Project* — In this £54-million project, a total of 100,047 files were collected on the server. 8,677 files (8.7%) were recorded in a database, and 91,370 files were stored in a file system. 4 files were issued to the client based on database search.

3. SYSTEM OVERVIEW

Table 3.1: Statistics of three Laing O'Rourke Projects– files captured and stored in distributed file system, files recorded in RDBMS, and the number of files issued to the customer. [Source: Laing O'Rourke]

Project	Office Building (£250m)	Education Building (£54m)	Large Infrastructure (£200m)
Total files generated	591,906	100,047	461,711
Files stored in file-system	546,906	91,370	446,173
Files stored in database	45,000	8677	15,538
Files issued to client	6566	4	2626

- *Large Infrastructure Project* — In this £200-million project, a total of 461,711 files were collected on the server. 15,538 files (3.4%) were recorded in a database, and 446,173 files were stored in a file system. 2,626 files were issued to the client based on a database search.
- *Office Building Project* — In this £250-million project, a total of 591,906 files were collected on the server. Out of them only 45,000 files (7.6%) were recorded in a database, and 546,906 files were stored in a file system. 6,566 files were issued to the client based on database search.

Less than 10% files are stored and hence searchable through databases and the rest are archived in distributed file systems.

Now we estimate the average time needed by an administrator to perform each tasks while considering a database, an enterprise search engine and a VAESE. It is to be noted that the administrators time is also spent on database, file system, network, and other related administration and maintenance activities. Administrators also need to spend their time on other official activities, e.g., training, meetings, discussions, appraisals, breaks, refreshments, responding to emails and so on.

Enter a record in the database. This is the estimated time to enter a file's records into a database table. The primary tasks involved are: (a) Extracting relevant information from a file's metadata, content, project details, customer details and so on. (b) Entering the relevant information of the file in a database. Based on the statistics about the three building projects, let us consider a scenario where 6.5% files (as the average of the three projects) are to be stored in a database. We consider a modest project where total 100,000 files are generated over a period of 2 years and 6,500 files (6.5%) are recorded in a database, i.e., 3,250 files per year. Assume that there is one administrator who is working fulltime for recording these files into a

database. When these tasks are spread over a year it is about **40 minutes** per task. As discussed before this includes overheads of a typical employee who as the file administrator. If all 100,000 files needed to be recorded in the database, we would need 15 administrators.

Capture and store a file. This is the estimated time required to capture and store a file in a distributed file system. The main tasks involved are: (a) Digital scanning of a file (if it was received in a hard copy), (b) Scanning of the contents or metadata of the file; it may take more time to open and scan a large file (e.g., a CAD file) than a small document. (c) Placing the file in an appropriate folder or sub-folder (if a folder or sub-folder does not exist then it should be created first). (d) Avoiding storing duplicates, or taking care of the multiple version naming. Assuming that 20 different fields (e.g., metadata, project details, client details, file access controls, etc.) to be addressed, we estimate that it will take **10 minutes** to perform this task.

Local search of a record in a database. As the files are recorded in the database with all the relevant information or fields, the search results are accurate. Therefore, the required search time is mainly the time required to populate the search form, which is estimated to be **2 minutes**.

Centralized search using a search engine. Searching for a relevant file stored in a large file repository using a search engine is a time consuming task. It is also a time consuming task to deal with the false positives and false negatives. Search results are presented as text in a list or tabular view ideally 10-15 results per page and a user often scans through 1-2 pages. We estimate the time to scan 10-15 search results and to identify a relevant file to be **15 minute**.

Centralized search using a visualization assisted search engine. Visualization assisted search can reduce the work load particularly in dealing with the false positives and false negatives. Moreover, some of the additional overhead time due to work leave can be avoided by improved collaboration among the administrator supported by provenance visualization. We estimate that the time can be reduced by around 50%, i.e., **8 minutes** per file.

Packaging. A constant amount of time is required to create manuals or disks with the deliverable, and for final quality checking of the deliverables. We estimate the time could be **2 minutes** per file.

3. SYSTEM OVERVIEW

Table 3.2: Statistics for life-cycle of files costs in person-days (8 person-hours per person-day). [Source: Laing O'Rourke]

Project	Office Building (£250m)	Education Building (£54m)	Large Infrastructure (£200m)
Database	61,712	10,422	48,117
Enterprise search engine	12,564	2084	9712
Visualization-assisted enterprise search engine	12,468	2084	9674

Table 3.3: Search-only cost in person-days (8 person-hours per person-day): local search in database, centralized search using enterprise search engine, and centralized search using visualization-assisted enterprise search engine. [Source: Laing O'Rourke]

Project	Office Building (£250m)	Education Building (£54m)	Large Infrastructure (£200m)
Database	27.36	0.02	10.94
Enterprise search engine	205.19	0.13	82.06
Visualization-assisted enterprise search engine	109.43	0.07	43.77

3.4.2 Life-cycle Cost

Using the *Office Building Project*, if entering a file record into a database takes 40 minutes on average, entering 45,000 files would have taken 30,000 person-hours or 3,750 person-days. If all 591,906 files had to be recorded in the database, it would take nearly 49,325 person-days. Table 3.2 presents the statistics for life-cycle costs. The cost for the database includes capture and storing in the file system, recording files in the database, local search in the database and, packaging. The cost for enterprise search engine includes capture and store in the file system, centralized search with the enterprise search engine, and packaging. Finally for VAESE includes capture and store in the file system, centralized search with the visualization supported enterprise search engine, and packaging.

3.4.3 Search Cost

The *Office Building Project* is used as an example. With a database, if one takes 2 minutes on average to find a file in the database to be issued to a client, 6,566 files would take about 219 person-hours or about 27 person-days. If one takes 15 minutes on average to find a file using a search engine, 6,566 files would take about 1,642 person-hours or about 205 person-days. If one takes 8 minutes on average to

find a file using a VAESE, 6,566 files would take about 875 person-hours or about 109 person-days. See the Table 3.3 for the statistics concerning other projects.

3.5 Summary

In this chapter, we first discussed three alternative solutions: databases, enterprise search engines, and Visualization-assisted enterprise search engine for enterprise data management and search. Thereafter, we presented an overview of the proposed VAESE system: the search engine modules, the user interface, and the visualization components.

We described a business costs estimation which was done in collaboration with our industrial partner. When considering the whole life-cycle of files, the benefit in terms of the possible time savings would be about 1-2% in comparison with search without visualization. However, when considering search effort only, visualization assisted search is expected to save some 30-50% of time in comparison to search without visualization. Therefore for a large construction enterprise, such a savings would not be trivial, particularly when it is translated into a monetary term.

It is thus highly desirable to use visualization-assisted enterprise search to reduce the costs involved. Visualization assisted search can reduce the work load particularly in dealing with the false positives and false negatives. Moreover, some of the additional overhead time due to workers on leave can be avoided by improved collaboration amongst the administrator supported by provenance visualization. All these suggest that databases may not be a winning strategy, though enterprise search engines also has shortcomings which can be improved by visualization capabilities.

4

Ontology-assisted Enterprise Search Engine

Contents

4.1	Introduction	63
4.2	Related Work	65
4.3	System Architecture	68
4.4	Multi-Ontology based Search	70
4.5	Experimental Results and Discussion	81
4.6	Conclusion	89

4.1 Introduction

In a large enterprise, there is typically a large volume of files distributed across different servers, network storages, and individual workstations. Although files may be organized one way or another, it is usually very difficult to have a protocol that can ensure all files are consistently named, all folders are properly placed, all duplications are avoided, all versions are precisely labelled, and all users who create, modify and read files follow the protocol to the letter. Hence, activities for search files are unavoidable, and in many cases there is no alternative that is more cost effective.

This work was motivated by a need in the construction industry for maintaining a large collection of files during each building project and for identifying a subset of these files for reporting purposes at the end of each project. As each building

project involves many stakeholders with different working practices, collecting all files from different stakeholders is one thing, but identifying a meaningful subset is a huge challenge. Effort has been made to deliver a solution using a database management system. This solution, however, suffers from a number of shortcomings, e.g., difficulties in implementing it across different stakeholders, costs of having numerous file administrators, and risk of single-point failure (the file administrator) in each project. Therefore, the use of computer-supported search provides a cost-effective mechanism for identifying a subset of files at the end of each project as long as all related files are stored. Although this work was motivated by such a specific need, the general need for enterprise-based file management is not uncommon. Hence, the approach that we employed to address such a need can potentially be deployed in many applications.

Numerous open source enterprise search engines (e.g., Solr¹, Elastic-Search², Xapian³, LucidWorks⁴, Terrier⁵ and so on) were developed for text-based document retrieval, among many of them are based on Lucene⁶ or Indri⁷. Lucene is a library providing a scalable index management facility [98]. Although the exact methods of commercial systems are unpublished, it is likely they use techniques similar to Lucene and Indri.

In the context of the industrial application, one major limitation of such enterprise search engines is that the document relevance in such content-based search is commonly measured with respect to some query (a set of keywords), and with the document treated as a bag-of-words [136]. In an enterprise, most of the files have a limited amount of textual data (e.g., spreadsheets, CAD, media, archive, images of building designs or construction sites, scanned copy of invoices and receipts, proprietary formats, etc.). Therefore, providing an improved retrieval mechanism based on metadata fields is essential [203].

In this chapter, we have developed a specialized enterprise search engine for supporting routine search operations to be performed by a team of administrators experienced in identifying relevant files in the reporting phase of projects. The system features a number of facilities, including an index, several dynamically

¹ lucene.apache.org/solr

² www.elastic.co

³ xapian.org

⁴ lucidworks.com

⁵ terrier.org

⁶ lucene.apache.org

⁷ www.lemurproject.org/indri

learned ontologies, and crawler agents. We mainly focus on the information retrieval components of the search engine, and present the ontology-based semantic support for the query expansion, ranking, and active learning. As it is not always easy for users to define search criteria accurately within one or a few attempts, it is necessary for the system to provide the following supports:

- (a) It is necessary to maintain an index, which can provide immediate responses to the users' query in an interactive fashion, and support collaborative search activities by multiple users.
- (b) It is necessary to complement (a) by using crawler agents to search beyond the index, and report to the users concerned as soon as any new file discovery is made. Together (a) and (b) enable the search activities scalable to a very large distributed file system.
- (c) It is necessary to improve the basic search criteria using the knowledge captured in the previous search activities. Such knowledge can be captured and stored in multiple ontologies, and can be used to expand a query which is particularly useful when different keywords are used to describe the same concept in the documents and in the queries and to prioritize search results, thereby improving the recall.
- (d) It is necessary to help users to identify false positives quickly through an effective visual interface and users' feedback about the true and false positives become part of the active learning process for updating ontologies dynamically as manually building such ontologies is a tedious and time consuming task. Active learning of the ontologies improves the relevance ranking of the search results over time. In addition to that the ontologies can also be tuned manually.

4.2 Related Work

4.2.1 Enterprise and Desktop Search

Recently there has been extensive reports on improving file search on desktops, e.g, Microsoft Windows Desktop Search, Google Desktop, Apple MAC OSX Spotlight, Zeitgeist on Ubuntu GNU/Linux. Keyword-based search for the discovery

of software has been proposed in [61,158,159]. Open source systems e.g., Lucene⁸, Indri⁹, and Xapian¹⁰ were developed for text-based document retrieval.

Personal Information Management (PIM) tools, e.g., SEMEX [35] and Haystack [114] allows users to organize their data semantically by creating associations between files and then leveraging these associations during search. iMeMex Data Model (iDM) [62] was proposed to represent unstructured, semi-structured as well as structured data inside a single model. Xu *et al.* [226] proposed a generic data model to capture and represent file semantics. Khoo *et al.* [120] reported that desktop users tended to prefer metadata, e.g., location, directories, dates, and names over keyword-based search. Teevan *et al.* [199] investigated the users' behavior when searching for e-mails, files, and the web. The latest work on PIM [211] implemented a multidimensional search approach that allows users to perform fuzzy searches for structure and metadata conditions in addition to keyword conditions.

The straightforward combination of field-level scores presented in works, e.g., [155] have several limitations, leading to efforts such as BM25F [174]. Phlat [55,74] developed an intuitive search interface to search files. Shah *et al.* [186] improved the quality of search by introducing the search context. Thomas [201] treated desktop search as meta-search problem for effective retrieval of individual collections. An adaptation of score combination and a smoothing method was suggested by [234] for the language modelling approach to information retrieval. Cohen *et al.* [54] improved the desktop search results by learning feature weights with training data.

4.2.2 Ontology-based Retrieval

Ontologies stores concepts and instances, along with their relations and properties as a knowledge-base [106]. Beagle++ [52] was developed for semantic desktop search. The ranking of documents according to an ontology was reported in [20,41,123]. Mylonas *et al.* [150] proposed a personalized information retrieval based on context and ontological knowledge. Vallet *et al.* used [207] ontologies and vector-space model to improve search over large document repositories. Other work on file systems, e.g., [44,95] enhanced the quality of file system search by leveraging the context of the search space. Castell *et al.* [41] proposed a retrieval

⁸lucene.apache.org

⁹www.lemurproject.org/indri

¹⁰xapian.org

model based on an adaptation of the classic vector-space model, including an annotation weighting algorithm, and a ranking algorithm. Fernandez *et al.* [81] proposed an ontology-based information retrieval model to support semantic search in a Web environment. ReFinder [59] is a context-based information re-finding system which assists users in re-finding their previously accessed files and web pages. A knowledge-based framework for integrating ontology-based personalized retrieval and reminiscence support was proposed in [189].

Leite & Ricarte [129] proposed multiple related ontologies for knowledge representation and integrated the system with a fuzzy model. A multi-ontology based multimedia annotation model was described in [65]. Vaidurya [144] is a document search system developed for clinical document search by using a domain specific ontology. A semantic ontology-based framework for personal information management was proposed in [225]. Query expansion techniques using ontologies can be found in [19, 60, 151].

4.2.3 Ontology Learning

Ontology learning techniques have been widely studied. A comprehensive review and discussion of major issues, challenges, and opportunities in ontology learning can be found in Zhou's survey [235]. Ichise *et al.* [108] proposed a framework for organizing the ontology mapping problem into a standard machine learning framework. This framework uses multiple concept-similarity measures. Maryam *et al.* [139] described the bottleneck of ontology learning while dealing with the knowledge acquisition and modelling domain knowledge. They presented a survey of the different approaches in ontology learning from semi-structured and unstructured data. Idrissi *et al.* [77] proposed a solution to overcome the manual knowledge acquisition bottleneck in ontology learning. They also presented a practical study of the methods that take data models as input to the learning process. Zhang *et al.* [230] proposed a semantic retrieval model of engineering domain knowledge, e.g., product design and process planning. They addressed the problems with existing keyword-based and semantic-enabled methods, and proposed an ontology-based semantic retrieval scheme for knowledge search and retrieval from domain documents.

The literature mainly focuses on content based retrieval, however, in an enterprise, most of the files are non textual (e.g., CAD, archive, audio, video, image, some proprietary formats, etc.). For that reason an improved metadata-based

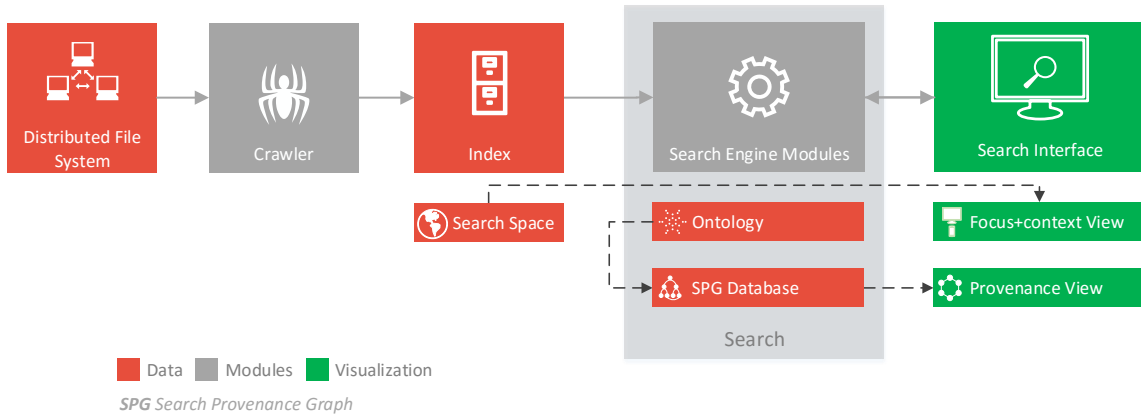


Figure 4.1: The architecture of the enterprise search engine. This chapter focuses on the *Search* part: multiple *Ontology* assisted (a) query expansion, (b) field-based ranking, and (c) active learning of the ontologies. The search engine also facilitates the functionalities (e.g., the search space information and ontologies for computing semantic query similarity in SPG) for the integration of visualization components to be discussed in following chapters.

retrieval is essential which is supported by an improved relevance ranking (e.g., precision and recall) [63] and machine learning [54] techniques.

4.3 System Architecture

We designed and prototyped an enterprise search engine to experiment with facilities that are not provided by conventional search systems. We optimized the speed and performance in finding relevant files for search queries by building a index of the files discovered by crawler agents. The main search engine consists of a collection of functional modules (*Search Engine Modules*, *Index*, *Crawler*, *Ontology*, and *SPG Database*), which reside between the *Search Interface* and the distributed file system, see Figure 4.1. Detailed description of the search process and the role of each functional module was discussed in Section 3.3.

Search Engine Modules. When a user initiates a search session by submitting a search query through the *Search Interface*, the *Query Processing Module* first parses the query and derives a set of search criteria. The criteria may consist of one or a few keywords and/or some constraints of file attributes, such as type, size and date as a filtering condition. The criteria are then enriched (see Section 4.4.2) by the *Ontology* module to expand the search space. For example, a keyword is enriched with similar words, and a file type is enriched with similar file types. The enriched

search query is then processed with the *Index*. This also activates the *Crawler* to find files that meet the criteria but are not in the *Index*. Because of the size and distributed nature of the file system, we anticipated that scanning the entire file system at an interactive speed is not feasible. Finally, the *Result Processing Module* ranks (see Section 4.4.3) the retrieved files before returning the results to the *Search Interface*.

Index. The search engine creates an index and the queries are executed to search through the index. The *Index* stores the information about the files and folders of the distributed systems (search space). In order to optimize the scalability of the *Index*, if required, it can only store the information about the files and folders that have been and may potentially be related to the recent search queries. In the *Index*, the *Index Management* module collects, scans, and parses the metadata of the files fetched by the crawler and creates a data structure which can be searched quickly. At first, the *Index Management* module analyzes each parsed word to remove stop words and to normalize the words. This stage optimizes the parsed result. Next, the *Index Management* module creates a data structure of inverted indexes. A detailed description of index data structure, and main operations and algorithms used for *Index Management* can be found in Appendix A.

Crawler. A crawler is a set of agents that scans through the distributed file system in an autonomous manner and at a predefined scheduled interval. The crawled information about files are used to update the *Index*. In an enterprise update, remove or creation of files happen very frequently. In order to keep the index fresh the crawler agents crawl the distributed file system sources every night. The frequency of crawling can also be reconfigured. As the crawler often discovers new files that met the search criteria of these queries, though it may take a few hours due to the size of the file system. These newly discovered files are updated to the *SPG Database* that can be viewed in the provenance visualization facility presented in Chapter 6.

Ontology. Ontologies are used for knowledge management. The system is equipped with multiple ontologies, including a term ontology, a type ontology, a size ontology, and a age (date-time) ontology. For example, the term ontology stores the basic terms in English (including abbreviations), and the weighted relationship between them. The type ontology captures the relationship between categories of file types and terms in English. The details of the ontologies are presented in Section 4.4.1.

Ontologies are used by the *Search Engine Modules* to expand a set of search criteria and to rank the search results. All four ontologies are automatically updated by an *Ontology Management* module upon receiving updated weights from *Active Learning* module. The users' actions of selecting files (i.e., true positives) serve as an active learning process (see Section 4.4.5) for the automated improvement of the ontologies.

The ontologies can also be manually updated or tuned by an administrator called *Ontology Manager* also known as *Relevance Engineer* who has a knowledge of the enterprise data using a graphical user interface. The Relevance Engineer identifies new concepts and the relationships that is relevant to search and update them to the ontologies. The Relevance Engineer also reviews the ontologies and manually tunes the weights of the ontologies to improve the relevancy of the retrieved documents.

SPG Database. A provenance database stores the queries performed by the users, the search results received, and the relevant results that are selected by the user. The search histories provides the provenance data to the *Provenance Visualization* facility.

Search Interface. The *Search Interface* is designed for two monitor displays of 3840x2160 pixel resolution. It has seven primary features, the first display shows (1) *Search Query Form*, (2) *Tabular View*, provenance visualization using (3) *Focus+context View of SPG*, (4) *Glyph-based View*, (5) *Collaboration Summary View*, and the dialogs for (6) *Late Discovered Files*, and the second display shows (7) *Treemap-based Focus+context View* of search space and search results. The details of the visualization facilities are described in the following chapters.

4.4 Multi-Ontology based Search

When a human assistant looks for files, his or her knowledge about various properties of files and their contexts plays an important role. When given a brief requirement for a search task, an effective assistant can narrow down a query or expand a query based on such knowledge. As our application involves a large volume of search activities on a daily basis, we can capture such knowledge using ontologies, use the knowledge to help human users in their search activities, and update the knowledge through active learning when the searches are performed.

Ontologies express knowledge in terms of concepts, their properties, and the relationships among concepts, and they are used in conjunction with a set of inference rules [34]. In our system, the search engine is supported by multiple ontologies (to be detailed in Section 4.4.1). When an input query is given, the search engine uses the ontologies and inference rules to expand a query to remove the necessity for users to specify search criteria precisely and comprehensively, and reduce the burden for users to try various combinations (e.g., different synonyms) in an exhaustive manner (to be detailed in Section 4.4.2). With the expanded query, the search engine usually finds many files. The search engine then uses the ontologies and inference rules to determine a relevance-score for each file returned. With the relevance-scores, the search engine is able to order the search results in the search interface, prioritizing users' attention for the most relevant files (to be detailed in Section 4.4.3). When users select files from search results as part of the intended collection, the selection interaction is utilized by the search engine seamlessly for active learning. The learning process infers that the selected files are considered as "true positives", and updates the weights in the ontologies accordingly (to be detailed in Sections 4.4.4 and 4.4.5).

4.4.1 Ontologies for Knowledge Management

At an early stage of this work, we found that using a keyword ontology such as WordNet [141] is not sufficient. We thus developed multiple ontologies to capture and update the knowledge about major attributes of files and the relationships between such attributes and the keywords. In addition to a Term Ontology, we have a Type Ontology, a Size Ontology and a Date Ontology. We implemented these ontologies using DEX/Sparksee [138], a scalable and fast C++ graph database, which can scale up to billions of objects [9, 137].

Term Ontology. The *Term Ontology* stores the knowledge about the terms or keywords used in the search and their relationships. It is modelled based on WordNet [141], with additional weights assigned to all edges. The original WordNet is an unweighted digraph, where each term is connected to set of synonymous terms called a synset. In file search, the words are not necessarily related to each other in exactly the same way as in spoken and written English. We thus introduce a weight to each directed edge, defining the level of "closeness" between two terms. Here the concept of closeness not only encompasses the basic notion of synonyms, but also captures the sense of co-occurrence in searches. For example, "building"

and “construction” represent the former, while “work” and “plan” represent the latter. Hence a query about one keyword (e.g., “building”) can be expanded to include similar words as well as common co-occurrences.

The structure of the term ontology Φ_K is essentially a digraph $\Phi_K = \{\mathbb{K}, \mathbb{W}\}$. $\mathbb{K}$ is a set of nodes that represent terms or keywords. $\mathbb{W} \subseteq \mathbb{K} \times \mathbb{K} \times [0, 1]$ is a set of directed edges. Each edge (u, v, w) defines the level of closeness from term u to term v with a weight $w \in [0, 1]$. All weights are initialized to 0.5 before any training (see Section 4.4.5).

Type Ontology. The *Type Ontology* expresses knowledge, e.g., term, and file types, their properties, and the relationships among them. It is a complete weighted undirected bipartite graph $\Phi_T = \{\mathbb{K}, \mathbb{T}, \mathbb{W}\}$, where $\mathbb{K}$ and $\mathbb{T}$ are two disjoint sets of nodes, representing keywords and classes of file types respectively. $\mathbb{W} \subseteq \mathbb{K} \times \mathbb{T} \times [0, 1]$ is a set of weighted edges. For every pair of nodes, $(u \in \mathbb{K}$ and $v \in \mathbb{T})$, there is an edge with a weight $w \in [0, 1]$.

The knowledge captured in the type ontology is the level of association between a term and a file type. For example, terms such as “building” and “plan” are closer to file types used by Computer-Aided Design (CAD) software, while terms such as “finance” and “budget” are closer to spreadsheet types. In a specific organization or enterprise, there is normally a strong correlation between the file names and file types. Such correlations can be effectively utilized to improve search efficiency and effectiveness. In this work, we categorized over 200 file types (e.g., txt, docx, pdf, xlsx, etc.) into 16 classes (e.g., Document, Spreadsheet, etc.). The weights are initialized to 0.5 and are then updated through active learning.

Size Ontology. Similar to the type ontology, the *Size Ontology* is a weighted undirected bipartite graph $\Phi_S = \{\mathbb{K}, \mathbb{S}, \mathbb{W}\}$, where $\mathbb{K}$ and $\mathbb{S}$ are two disjoint sets of nodes, representing keywords and classes of file sizes respectively. $\mathbb{W} \subseteq \mathbb{K} \times \mathbb{S} \times [0, 1]$ is a set of weighted edges. The weights are initialized to 0.5 and are then dynamically updated through active learning.

This ontology captures the knowledge about the level of correlation between terms and file sizes. In order to support discriminable variations at the exabyte level as well as the kilobyte level, we divided file sizes into eight classes, namely 0-999 bytes, 1-999 kB, 1-999 MB, 1-999 GB, 1-999 TB, 1-999 PB, 1-999 EB, and 1 ZB or above.

Date Ontology. The *Date Ontology* is also a weighted undirected bipartite graph, denoted as $\Phi_D = \{\mathbb{K}, \mathbb{ID}, \mathbb{W}\}$, where $\mathbb{K}$ and $\mathbb{ID}$ are two disjoint sets of nodes, representing keywords and classes of file “ages” respectively. $\mathbb{W} \subseteq \mathbb{K} \times \mathbb{ID} \times [0, 1]$ is a

set of weighted edges. The weights are initialized to 0.5 and are then dynamically updated through active learning.

In a large organization or enterprise, the convention of naming files often reflects some historical protocols, and this knowledge can thus be used to enrich queries. The date/time attribute in a file system is normally displayed using one of the international or regional formats (e.g., YYYY-mm-DD_HH-MM-SS). In the context of file searching, it is more useful to use the age of a file, i.e., the interval between the time of searching and the record time of creation (or last modified/accessed). The main challenge is the large range of values. We thus adopted a convention similar to that for size, dividing file ages into five classes: hour (0-24 hours), day (1-31 days), months (1-12 months), year (1-10 years), and decade (1 decade or more).

The search engine is designed in such a way that it can easily integrate more ontologies, e.g., domain ontology for domain specific semantic support, personalized ontology for personalized information retrieval and so on.

4.4.2 Query Expansion

Query expansion reformulates the users' query by adding closely-related terms to the query. It enriches an input query to accommodate ambiguities in natural language, and alleviate the users' burden to define all these related terms explicitly. This feature is particularly important for an enterprise where the administrators need to find all files related to a project but are unfamiliar with the precise meta-information about these files (e.g., the naming convention). With query expansion, the search engine improves the recall in the search results.

The expansion procedure is invoked by the *Query Processor* upon receiving a query. The original query is parsed, and a set of terms (or a term vector) is derived. For an input term "building", for example, it would also be useful to search for other words, such as "construction", "build", "make", "progress", "establish", etc. as files related to buildings may not be saved with file names containing the word "building", especially in a project all about buildings. This set of terms can further be enriched recursively. In our trained term ontology, for instance, terms such as "structure" and "progress" are two links away from the term "building".

In the context of the term ontology, each input term is referred to as a source node, and each added term as a destination node. Given one or a few input terms in the original query, the expansion procedure first locates the corresponding

source nodes in the term ontology. Starting from each source node, it activates a recursive sequence for determining a *knowledge spanning tree*, connecting all nodes with an appropriate weight, and reaching out to all its destination nodes. The connectivity is determined by an aggregate weight $W(s, d)$ between a source node n_s and a destination node n_d . Let $w_1, w_2, \dots, w_k$ be the weights on the k edges connecting n_s and n_d in the term ontology. The aggregate weight $W(s, d)$ is computed as follows:

$$W(s, d) = \prod_{i=1}^k w_i \rho(i)$$

where $\rho(i)$ is a step function that moderates the impact of the weight according to connectivity between node n_s and edge i . A commonly-used step function is $\rho(i) = i^\alpha, \alpha \leq 0$. As $i \in [1, k]$ is an integer variable, the function appears to be step-down or a horizontal (when $\alpha = 0$). We set $\alpha = 0$, i.e., $\rho(i) \equiv 1$, in this work.

Alternative step functions may also be used. For example, to compute $W(s, d)$ using only the notion of synset in WordNet, one would have $\rho(i) = 1$ if $i = 1$ and $\rho(i) = 0$ otherwise. To consider all descendants which are k links away from n_s , one would have $\rho(i) = 1$ if $i \leq k$ and $\rho(i) = 0$ otherwise. To consider all descendants in the ‘is-a’ hierarchy, one would have $\rho(i) = 1$ if edges, $1, 2, \dots, i$ are all “is-a” connections, and $\rho(i) = 0$ otherwise.

$W(s, d)$ is used to control the depth of the spanning tree. The recursion stops when $W(s, d)$ becomes insignificant, i.e., less than a threshold value. In this work, the threshold is set as 0.25. For all input terms in the original query, the knowledge-spanning trees define the expanded set of terms to be used in the search, facilitating meaning-based retrieval based on the semantic representation of the original query in the term ontology.

4.4.3 Ranking

The search engine returns search results in two batches. The results in the first batch are derived by the search engine from the index and are returned at an interactive rate. The results in the second batch are delivered by crawler agents, which search the entire file system to discover files that match with the expanded query but not in the index. Because the size of the file system is large, the results in the second batch (if any) usually cannot be produced instantly, and are delivered to users gradually in a manner similar to a message box. For both batches, the search engine provides each resultant file with a score representing the level of relevance

to the original input query. For the first batch, these scores are particularly useful in ordering the display of the search results, so users can pay attention to files that are likely to be most relevant. For the second batch, the results are normally displayed in a temporal order (first come and first served). The scores are thus supplementary. They are used to order the display only when there are many results coming at a similar time, or where users prefer to process them in big groups.

Each file in a batch of results has a set of attributes as the metadata of the file, e.g., file name, type, date (creation/last modified/accessed), size, owner, group and so on. These attributes can be used to rank the file with a relevance score. In this work, we implemented a ranking function based on file path, type, date and size, corresponding to the four ontologies described in Section 4.4.1.

Let Θ be a set of search terms $\{\theta_1, \theta_2, \dots, \theta_n\}$ in the original query. Let F be a collection of files in a batch of search results. Each file, $f \in F$, has t attributes. We first decompose a file path into a set of terms available in the term ontology. For example, for a file path “/Cost Management(dsfp05)/Insite Server/Whole Life Cost/Network Rail/Projects/17003 Construction Budget v0.1.xlsx”, the decomposition results in a set of search-able terms $\{17003, \text{construction}, \text{budget}, \text{v1.0}, \text{project}, \text{network rail}, \text{whole}, \text{life}, \text{cost}, \text{insider server}, \text{cost}, \text{management}, \text{dsfp05}\}$. We denote such a set of terms derived from a file path by $\Psi = \{\psi_1, \psi_2, \dots, \psi_m\}$.

As described in Section 4.4.2, given a source term (root) and a destination term (non-root) in the same spanning tree, we can compute an aggregate weight $W(s, d)$. Hence for every pair of terms, $\theta_i \in \Theta$ and $\psi_j \in \Psi$, if ψ_j is in the spanning tree built for θ_i , we can obtain an aggregate weight $W(\theta_i, \psi_j)$. Otherwise, $W(\theta_i, \psi_j) = 0$. For each θ_i , we define a composite weight

$$W_W(\theta_i) = \max_{j=1}^m W(\theta_i, \psi_j)$$

While this composition is normally sufficient, the following iterative composition function can be used to reflect, more accurately, the partial additive nature of the composition.

$$\begin{aligned} \kappa(1) &= W'(\theta_i, \psi_1); \\ \kappa(j) &= \kappa(j-1) + (1 - \kappa(j-1)) W'(\theta_i, \psi_j); \\ W_W(\theta_i) &= \kappa(m) \end{aligned}$$

where the series of values $W'(\theta_i, \psi_j)$, $j = 1, 2, \dots, m$ is obtained by sorting $W(\theta_i, \psi_j)$ in descending order.

In addition, for each file $f \in F$ and each term θ_i in the original query, we can locate a connection between θ_i and the file type class of f in the type ontology, and obtain a weight assigned to that edge. We denote this weight as $W_T(\theta_i)$. Similarly, we obtain a weight $W_S(\theta_i)$ for the size of f and another weight $W_D(\theta_i)$ for the date of f .

We then combine $W_W(\theta_i)$, $W_S(\theta_i)$, $W_T(\theta_i)$, and $W_D(\theta_i)$ as

$$CW(\theta_i) = 1 - \prod_{x=W,S,T,D} (1 - W_x(\theta_i))$$

Finally, for each file $f \in F$, we compute a score based on all terms in the original query as:

$$S(f) = \max_{i=1}^m CW(\theta_i)$$

Similar to the computation of $W_W(\theta_i)$, a more accurate but costly composition function can also be used for $S(f)$. The scores for all files in a collection of search results F provides a means for sorting these files in descending order in their display, enabling users to prioritizing their attention and effort cost-effectively.

4.4.4 Relevance Feedback

When the search results are displayed to the user, the user uses the visual interface to select those files that are considered to have met the purpose of the search activities, e.g., they are required by the reporting process for a construction project. The user's selection actions implicitly provide some information about the relevance of the files in the search results. This information can be used as part of active learning to improve the ontologies dynamically.

However, while a selection action can be treated as a positive confirmation of relevance, the fact that a file is not selected cannot be treated naively as a confirmation of irrelevance. A search often returns many files. A user may feel overwhelmed after selecting a few files, or just viewing the results, and decide that it is better to reformulate the query. Taking such scenarios into account, we derive a set of integer values, Λ , from a user's selection activities after receiving search results before moving on to a new query process. Each value λ in the set Λ is referred to as a *Relevance Feedback Index* (RFI), and each corresponds to an edge

in one of the ontologies. It is used in the Active Learning process for improving the ontologies.

Using the same notation from Section 4.4.3, we consider the term ontology first. Consider a set of files returned by a search $F = \{f_1, f_2, \dots, f_s\}$. The user's selection activities result in a binary flag ι_k for each file f_k such that $\iota_k = 1$ if f_k has been selected and $\iota_k = 0$ otherwise.

For each input term, $\theta_i \in \Theta$, in the initial query, we consider all ontological nodes that are featured in F and are related to θ_i as a generalized set Φ_i . In other words, this set Φ_i includes the file type classes, file size classes and file dates classes of all files in F . It also includes the union of all term sets Ψ_k , $k = 1, 2, \dots, s$, each derived from a file path. For each pair of ontological entities (θ_i, ϕ_j) , we need to derive an RFI, λ if there is a connection between them in a spanning tree where θ_i is the root and ϕ_j is a non-root node. This RFI can later be used to update the weight(s) along the connection.

We adapt the guideline proposed by Rocchio [175, 181] to compute the aggregated relevance score by combining both positive and (potentially) negative relevance feedback information. With Rocchio's method, positive and negative information are combined together to modify a query in the vector space model. Let F_s and F_n be two disjoint subsets of F , representing the selected and unselected files respectively. Given (θ_i, ϕ_j) , we first calculate two numbers, $s_{i,j}$ and $n_{i,j}$. $s_{i,j}$ is the number of times that ϕ_j (e.g., a type class) has been featured in F_s . $n_{i,j}$ is the number of times that ϕ_j has been featured in F_n . For each θ_i , we place all $s_{i,j}$ into a vector $\vec{S}_i$, and all $n_{i,j}$ into $\vec{N}_i$. We can then compute a set of RFIs as:

$$\vec{\Lambda}_i = \beta \vec{S}_i - (1 - \beta) \vec{N}_i \quad (4.1)$$

where $0 < \beta < 1$ is scalar constant that defines the ratio of the reward and penalty factors. In other words, it controls the influence of positive and negative relevance feedback.

In typical search scenarios, a small fraction of files among the search results would be selected by the users. Equation (4.1) indicates that either a low score of $\vec{S}_i$ or a high score of $\vec{N}_i$ would result in a low score of $\vec{\Lambda}_i$. In general $|F_s| \ll |F| \implies s_{i,j} \ll n_{i,j}$. As the difference between $s_{i,j}$ and $n_{i,j}$ can vary dramatically, it is difficult to set an appropriate β . We thus modified Equation (4.1) by mapping the values in the two vectors $\vec{S}_i$ and $\vec{N}_i$ into percentages. We thus divide each vector into four sub-vectors, corresponding to the four ontologies described in Section 4.4.1.

We denote them as $\vec{S}_{x,i}$ and $\vec{N}_{x,i}$ where $x = W, S, T, D$. The revised formula for computing the corresponding $\Lambda_{x,i}$ is given in Equation (4.2).

$$\vec{\Lambda}_{x,i} = \left(\beta \frac{\vec{S}_{x,i}}{\sum_{t=1}^{|\vec{S}_{x,i}|} s_{x,i,t}} - (1 - \beta) \frac{\vec{N}_{x,i}}{\sum_{t=1}^{|\vec{N}_{x,i}|} n_{x,i,t}} \right) \times 100 \quad (4.2)$$

where, $s_{x,i,t} \in \vec{S}_{x,i}$, and $n_{x,i,t} \in \vec{N}_{x,i}$. As per the suggestions in the existing literature [136], we have empirically set the value $\beta = 0.65$ based on the pilot studies conducted with several datasets (between 0.5 to 2.0 millions of files). The four sets of RFIs, $\vec{\Lambda}_{W,i}$, $\vec{\Lambda}_{S,i}$, $\vec{\Lambda}_{T,i}$ and $\vec{\Lambda}_{D,i}$ are forwarded to the learning process to update the four ontologies.

4.4.5 Training and Active Learning

The weights in the four ontologies were initialized to 0.5 at the very beginning. They have all undergone a period of training in laboratory conditions. However, this is not sufficient to accommodate the dynamics in a real word environment. We thus implemented a mechanism for active learning, which utilizes the relevance feedback indices (RFIs) described in the previous section.

Each RFI, λ , is rounded to an integer. A positive value indicates that the weight(s) of the corresponding edge or edges along a path should increase to reflect the positive confirmation of relevance. A negative value indicate that the weight(s) should decrease. We use $|\lambda|$ to determine the number of repeated application of small increments or decrements. We have adopted the update mechanism commonly used in neural networks [21]. When each edge is modelled as a simple neural network, the small amount of increment or decrement depends only on the current weight w after the factor of λ is taken care of. Given a weight w on an edge, the small increment (or decrement) is computed in the following steps.

We first convert the weight w in an ontology to a weight v in a neural network as:

$$v = -\ln\left(\frac{1}{w} - 1\right)$$

Here we define the logistic function as:

$$w = f(v) = \frac{1}{1 + e^{-v}}$$

While η is a constant for moderating the scale of increment (or decrement), the amount of increment (or decrement) is then computed as:

$$\begin{aligned}\Delta &= \eta \delta \frac{\partial f(v)}{\partial v} \\ &= \eta \delta (f(v) (1 - f(v))) \\ &= \eta \delta w (1 - w)\end{aligned}$$

$$\delta = \begin{cases} 1 - w & \text{if } \lambda > 0 \\ -w & \text{if } \lambda < 0 \end{cases}$$

The neural network weight v' is computed and then converted back to an updated ontological weight as:

$$\begin{aligned}v' &= v + |\lambda| \Delta \\ w' &= f(v') = \frac{1}{1 + e^{-v'}}\end{aligned}$$

A simplified form w' which is computed from w can be derived as:

$$w' = \frac{1}{1 + \left(\frac{1}{w} - 1\right) e^{-|\lambda| \eta \delta w (1-w)}} \quad (4.3)$$

4.4.6 Weight Normalization

During the active learning process the weights of the concerned edges keep changing, which also changes the average weight of the ontologies. In order to conserve the average weight of the ontologies as a constant (which is here set to 0.5), a house keeping routine periodically normalize the weights at a predefined scheduled interval. The regular normalization function is described in Algorithm 4.1. The function first determines a scaling factor α , and then scales the weights linearly. However, this does not guarantee that the scaled weights will be between $[0, 1]$. Therefore, if the weight of an edge overshoots 1.0 then a nonlinear scaling factor k is computed which scales down the weight. This process continues recursively until the average weight becomes 0.5 and the individual weights are set to $[0, 1]$.

Algorithm 4.1 NormalizeWeights (w'_i)

Description: Normalization of the directed graph weights.**Require:** w'_i is an array of n elements or updated weights after active learning.

```
1: function NORMALIZEWEIGHTS( $w'_i$ )
2:    $n = |w'_i|$  ▷ Cardinality
3:    $\mu = \frac{1}{n} \sum_{i=1}^n w'_i$  ▷ Mean
4:   if  $\mu = 0.5$  then
5:     return  $w'_i$ 
6:   else
7:      $\alpha = \frac{0.5}{\mu}$  ▷ Linear scaling factor
8:     for all  $w'_i$  do
9:        $wn_i = \alpha w'_i$  ▷ Scale
10:    end for
11:     $max = \text{Max}(wn_i)$  ▷ Find the maximum weight in the array
12:    if  $max < 1$  then
13:      return  $wn_i$ 
14:    else
15:       $c \mid 0.5 < c < 1$  ▷ Choose  $c$  between 0.5 and 1
16:       $k = \frac{\ln(c+1)}{\ln(max+1)}$  ▷ Non-linear scaling factor
17:      for all  $wn_i$  do
18:         $wn'_i = (wn_i + 1)^k - 1$  ▷ Scale
19:      end for
20:    end if
21:  end if
22:   $\text{NORMALIZEWEIGHTS}(wn'_i)$  ▷ Recursive call
23: end function
```

Figure 4.2 shows an example snippet of the *Term Ontology*. It illustrates how the weights of the outgoing edges of the term “building” to its related terms one-link away improve with training. The source term “building” is connected to nine different terms and itself. Four relevance feedback datasets related to the search term “building” are used from the *SPG Database* database for training purpose. The initial weights are set to 0.5. For each training the RFI values λ , new weight w' , and the normalized weight wn are computed. Here, the values of $\beta = 0.6$ (Equation 4.2), $\eta = 0.3$ (Equation 4.3), and $c = 0.6$ (Algorithm 4.1) are considered for the computation.



Figure 4.2: A snippet of the term ontology shows the weights of the search term “building” and its related terms in one-link away. In this example, four different sets of users feedback data are used to train the ontologies, showing the RFIs and changes in weights before (w') and after normalization (wn).

4.5 Experimental Results and Discussion

4.5.1 Experimental environment.

All experiments were performed using the prototype enterprise search engine that was implemented in C++. Experiments were run on a Linux 3.11.0 kernel installed on a virtual machine(VM) running on a 64-bit hyper-threaded 2.8 GHz Intel Xeon processor, 6 GB of memory, and a 7,200 RPM 500 GB SCSI disk. The VM’s operating system was configured to use one processor and 3GB base memory.

4.5.2 Data Set

Over the years, several standard methods have been evolved to evaluate information retrieval and search systems. TREC provides a set of files collection, queries and relevance judgement information for evaluating web information retrieval systems [81,82]. Task-based and focus groups (or log analysis) studies were conducted by gathering a set of real users and have them interact with the system over

Table 4.1: Size distribution for different types of files

File Category	Total Files Analyzed	Distribution Type	Distribution Parameters
Audio	327	Weibull	$\alpha = 0.63017, \beta = 1.4268 e^5$
Backup	113	Lognormal	$\sigma = 2.7162, \mu = 11.079$
Compressed	121	Lognormal	$\sigma = 3.2318, \mu = 12.623$
Database	597	Lognormal	$\sigma = 1.5931, \mu = 9.1686$
CAD	56	Lognormal	$\sigma = 1.2267, \mu = 7.9307$
Developer	1550	Lognormal	$\sigma = 1.2497, \mu = 7.9466$
Disk Image	106	Weibull	$\alpha = 0.30197, \beta = 2.6857 e^5$
Document	8749	Lognormal	$\sigma = 3.1152, \mu = 8.8189$
Spreadsheet	223	Lognormal	$\sigma = 3.1372, \mu = 8.8102$
Presentation	34	Lognormal	$\sigma = 3.0591, \mu = 8.8049$
Executable	4183	Lognormal	$\sigma = 1.7659, \mu = 11.783$
Image	13138	Lognormal	$\sigma = 2.0204, \mu = 8.527$
Misc	138	Lognormal	$\sigma = 1.2929, \mu = 14.421$
System	33332	Weibull	$\alpha = 0.54405, \beta = 2.5418 e^5$
Video	3372	Lognormal	$\sigma = 2.8214, \mu = 15.697$
Web	4809	Lognormal	$\sigma = 1.331, \mu = 8.2418$

some period [49, 54, 59, 78]. The ontology-based web information retrieval systems used news web archive [41, 189]. The latest work on enterprise search and PIM by Wang *et al.* [211] used their desktop datasets for evaluation. They used 43GB desktop user data consisting of 24K files of different types.

Performance evaluation of an enterprise search system using a large-scale dataset is a challenge due to unavailability of real file system collections or synthetic datasets and benchmarks due to privacy concerns [121, 211]. In order to address this problem, Chernov *et al.* [50] attempted to create a test bed for desktop search by collecting files and queries collaboratively. The recently created pseudo desktop collection by Kim & Croft [121] could be a useful dataset to evaluate our system, however the size of this collection is less than 200MB.

At the beginning phase of the project, we had meetings with members of the industrial partner, where the existing document management system and the requirements of the document search system were discussed. Thereafter, the author of this thesis had a week long placement in the enterprise. During that placement, a series of meetings with the domain experts were held, and had the opportunities to familiarize with the existing document management systems, the distributed file system, the file naming convention and directory structures, different file types,

the classification techniques used, and typical search scenarios. The requirements, the use cases, and the technical changes of the system were also discussed.

The primary dataset contained files and directories of total 1.7TB. This dataset contains 70,848 files organized in 7,338 directories. 24% of the files are multimedia files (e.g., audio, video and pictures), 12% document files (e.g., pdf, text, word, etc.), and 47% files are other types of files. More details of the different types of files analyzed can be found on the Table 4.1. The average directory depth was 3.4 with the longest being nine. On average, each directory contains 11.6 sub-directories and files, with the largest folder containing e-mails containing 1,013. The system extracted 347,448 unique stemmed content terms. File modification dates span seven years. 45% percent of the files are smaller than 177KB, and 65% of the files are smaller than 4.1MB. Files such as virtual machines images are as big as 60GB in size.

Based on the knowledge gained during the placement and the available dataset, a large simulated file system dataset is generated. The techniques proposed by Chernov *et al.* [50] and Kim & Croft [121] suggesting the usage of statistically generated dataset is adapted. The method proposed by Agrawal *et al.* [4] to statistically generate file system dataset is used for the evaluation of our system.

File system data contains different metadata fields, e.g., name, type, size, date, user, group, etc. From the existing dataset, we have derived different distributions for each of the metadata variables. For example, the file size follows a certain distribution, which can be confirmed from the histogram or distribution fitting tools. We use these distributions to create files of any scale randomly. The generated files distribution are validated with the original distribution using Kolmogorov-Smirnov(KS) test for the equality of continuous distributions. A file creation follows the following steps.

File System Structure. Engineering enterprises (e.g., Laing O'Rourke) deals with very large projects generating files in the scale of billions stored in a file system distributed across many offices located in different regions across the globe. The distributed file systems usually span into continents and regions which contain the files classified into different business unit owners, functional areas, projects and so on. Under such structure, different users create different folders to store and share the information. A statistics of distributed file system structure, hierarchy, and size is created based on the knowledge gained during the placement. Firstly, a directory name-space structure is defined in an XML file. Each XML node is assigned a weight, for example, Laing O'Rourke stores nearly 50% of their data in the United

Kingdom offices. Then the depth of the directory hierarchy is generated randomly. The name of the directory at each level is drawn iteratively from the probability mass function (PMF) derived from the XML file.

File Name. In an organization, file names commonly follow a convention such as “prefix_project-code_keyword(s)_version_suffix.type”. However, such naming convention is not followed consistently and users rarely memorize the project code to search for project specific files. The users mostly rely on keywords to search for files. For that reason it is sensible to generate the file names randomly. We have created files with different names by randomly drawing from a term distribution created from the existing dataset as well as a short dictionary containing most commonly used words, thereby increasing the diversity in the randomly created file name.

File Type. We have created a discrete probability distribution of file types by scanning the dataset. For each type, we calculate the weight or frequency by number of times they appeared in the data.

File Size. Different types of files exhibit significantly disparate size distribution [79], for example Agrawal *et al.* [4] reported that file size by count follows hybrid distribution of Lognormal and pareto-tail. Therefore, we have categorized files into 16 different high-level categories, and each category has subcategories containing more than 200 different known file types. We create separate distributions for each category which is presented in Table 4.1.

File Date. In the context of file searching, it is more useful to use the age of a file, i.e., the interval between the date of searching and the record date of creation (or last modified/accessed). We have created a continuous probability distribution of the file age from our existing dataset. We draw the age value from the age probability distribution and subtracted from the current date to assign a realistic date to the newly created file.

4.5.3 Experimental Results

Relevancy. In order to appreciate how well the search engine performs, i.e., learns to rank the search results, it is desirable to compare the performance under following conditions (similar to method used in [150,152]):

1. Without active learning of the ontologies where the relevant results expected to be distributed randomly in all the search results.

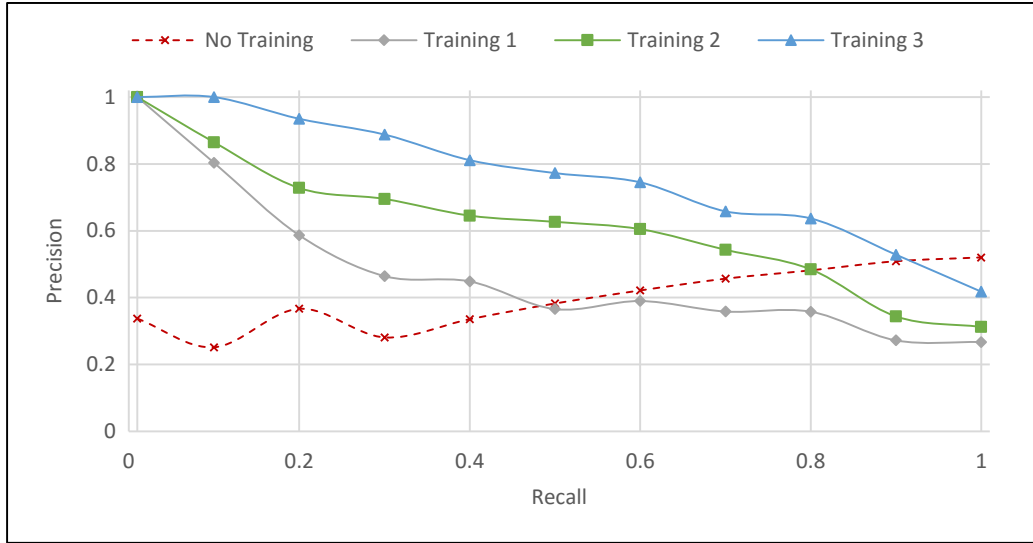


Figure 4.3: Comparison of the precision and recall before and after training with three different training sets.

2. With iterations of active learning using the relevance feedback information, where after each iteration the improvement of the relevance rank is expected.

We evaluated the effectiveness of the search on the basis of how well it helps users achieve their file searches efficiently and effectively. We focused on the searches within the *Index*, since this offers search results at an interactive rate. Hence the freshness of the *Index* is also particularly important. For a given query end-user perception about relevancy varies. In this context, we used the following performance measures: (a) precision reports the fraction of files in the results that are labelled as relevant, and (b) recall value (sensitivity) reports the fraction of relevant instances that are retrieved. By default, we consider that the results are ranked with respect to the ranking function described earlier.

Interactive relevance feedback gives very substantial gains in retrieval performance. Empirically, one or two round of training using relevance feedback is often useful. However, there is some subtlety to evaluating the effectiveness of relevance feedback. Relevance feedback requires enough judged files, otherwise the process is unstable in that it may drift away from the users' information need. Following the approach proposed in [211], we start with the initial set of 40 different synthetically generated queries for relevance judgement and train the ontologies. Following each search round we compute the average precision-recall graph for

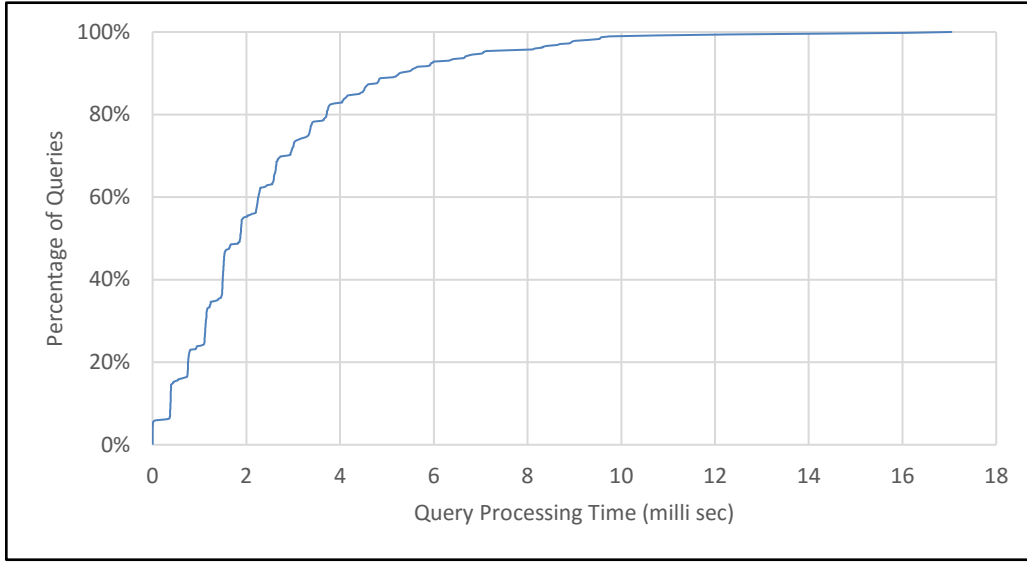


Figure 4.4: The CDF of query processing time, 500 queries over 0.5 million files.

all the queries while keeping the judged documents the same. We have measured higher performance after three consecutive training steps.

Figure 4.3 presents the recall, and mean average precision (MAP). Computing precision is very time-consuming, requiring the manual inspection of each file returned for each query and deciding whether the file is relevant to the query. Therefore, recall and MAP are averages across the entire sets of 40 different queries. In standard information retrieval, relevance assessment for every file against every query is given to compute the recall metric, which is not available in our case. Therefore, we measure the recall as the percentage of time that the target files are returned as one of the answers.

Figure 4.3 shows that at the beginning when the ontologies are not trained the relevant files are distributed randomly across all the returned files. After we start to train the ontologies using relevance feedback information, the relevant files starts to show up on the top. The precision and recall values show that the system ranks the relevant results effectively. Specifically, after training the search achieves higher precision. As it can be seen in the plot, the MAP at 0.10 recall is 0.80, 0.86, and 1.0 for three consecutive trainings. The plot shows that after each iteration of training the recall value improves. The ontology manager can also manually review and tune the weights of the ontologies which may further improve the precision and recall values.

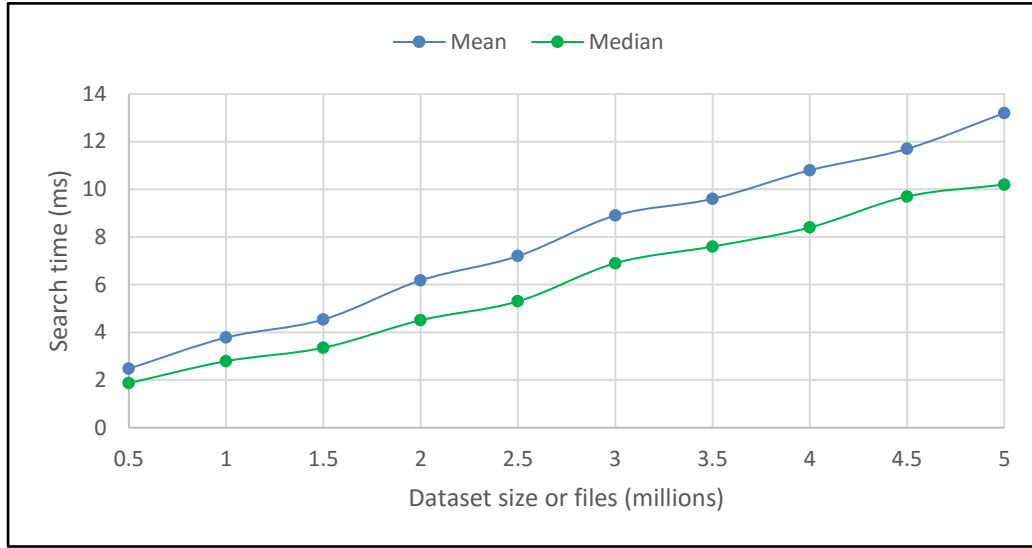
Query Processing Performance. Figure 4.4 shows the Cumulative Distribution Function (CDF) of the query processing time for 500 queries over a cache of 0.5 million files. The reported query processing times are averages of 50 runs. We observe that, the maximum query processing time is below 9.5 millisecond. Further, approximately 60 percent of the queries execute within 2.1 millisecond, while 95 percent of the queries execute within 6.5 millisecond. This is particularly important in term of user experience as high variance in response time is known to significantly degrade the user perceived usability of the system. Further analysis shows that the difference in query processing time is observed due to some queries are enriched with a large number of additional terms compared to others.

System Scalability. With an initial hardware configuration, the scalability is measured in terms of (a) the number of files the system can index or the size of the index with the number of crawled files, and (b) the query processing time over the index. The size of the ontologies should also be considered for the scalability measure.

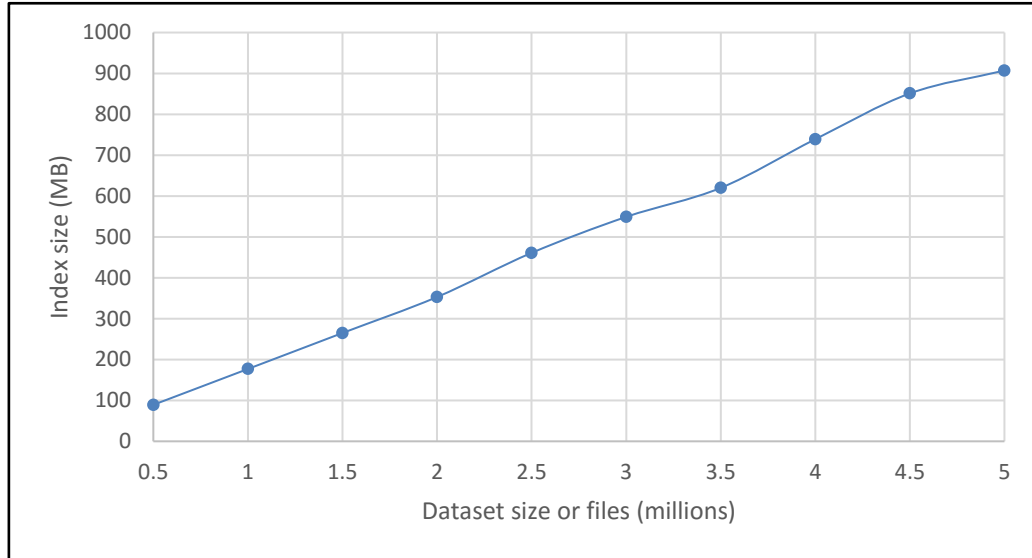
We believe that our experimental datasets are sufficiently large, given that the system is tested in a virtual machine. Therefore, the results apply directly to a large-scale distributed system. We briefly study the scalability of the system to assess its potential to handle very large datasets. Figure 4.5 (a) plots average and median query times for the same set of 500 queries against dataset size. The results show that query performance scales linearly with dataset size but with a relatively flat slope (e.g., increase of only 1 seconds in mean query processing time when the dataset increases by 0.5 million in size).

Figure 4.5 (b) shows the index size against dataset size or the number of files indexed. It shows that the index size scales linearly with dataset size and almost doubles as the dataset size doubles. The index is compressed with basic compression algorithms, as a result 0.5 million file system metadata are cached in 89.3 MB space, and 5 million files are cached in 907 MB space. This is a quite promising result because there are many known compression techniques that can be applied to improve the index size.

The ontologies are scalable in terms of the number of objects (nodes and edges) they contain. The term ontology has 58,094 unique term nodes connected to their corresponding synsets. The size ontology has around 58,102 nodes and 4,64,752 edges, so as the date ontology. The type ontology has 58,110 nodes and 9,29,504 edges. The ontologies can also be scaled to handle a large number of objects so that new concepts and relationships can be dynamically added either manually or



(a)



(b)

Figure 4.5: (a) The mean and median of execution times for queries vs. dataset size. (b) The index size vs. dataset size.

automatically.

This result is quite promising considering that our prototype has not been optimized for any dimensions other than structure, and, overall, has not been code optimized. There are many known optimizations that we can apply to improve the processing time. The test is conducted in a virtual machine installed on a desktop, given a host machine the performance would improve drastically. In order to improve the system scalability, the active learning process can be done by a house-

keeping process in a predefined scheduled interval particularly when the load on the system is very low.

4.6 Conclusion

In this chapter, we described an enterprise search engine developed for a large-scale distributed file system search in an engineering enterprise. In particular, we used multiple ontologies for query expansion, and ranking of search results. Our empirical studies proved the important role of ontology that the search engine relied on to achieve an interactive performance in search. The use of query expansion and active learning of the ontologies improved both recall and precision of the search query. As the studies confirmed that training plays an important role in improving the relevancy of the search results, they indirectly suggested that active learning will be necessary and effective in practical deployment. The empirical studies also showed that the system is scalable even with a virtual machine running in a desktop computer.

In the next chapter, we present a focus and context visualization technique to assist administrators to identify false positives, false negatives, and to support efficient multivariate search.

5

Search Result Visualization

Contents

5.1	Introduction	91
5.2	Related Work	94
5.3	Design Considerations	97
5.4	Visual Mapping of Glyph	100
5.5	Cyclically Animated Glyph (CAG)	103
5.6	Pre-development Empirical Studies	106
5.7	Post-development Evaluation	113
5.8	Conclusion	116

5.1 Introduction

This chapter is concerned with *focus and context visualization*. In particular we address a requirement for displaying search results represented by small visual objects as the focus are discretely distributed on top of a search space represented as a dense visual context. Many visual representations, such as treemaps [111], pixel-based visualizations [118] and images, make highly cost-effective use of space in conveying complex information. However, when these representations are used to depict contextual information in a focus and context visualization, it is difficult to overlay focal information on top of the context while facilitating efficient visual search.

For example, Figure 5.1 shows a zoomed-in view of a treemap that represents part of a GNU/Linux file system. In many situations, it is useful to display search

results (i.e., *focus*) in conjunction with their distribution in the file system (i.e., *context*). Such a visualization can help users determine if search criteria were set correctly or if the search has resulted in any unexpected outcomes, i.e., *false positives* and *false negatives*. In Figure 5.1, we can observe that when the search results are displayed as static coloured dots against a treemap background, it is not easy to identify these dots. With the rapid growth of data stored in distributed file systems, the level of difficulties in, and the required effort for, searching files will likely increase. It is becoming highly desirable to provide a search interface with effective focus and context visualization.

In Chapter 4, we designed and prototyped an enterprise search engine, enabling the experimentation with visualization facilities, which could not be done with a blackbox search system. In this chapter we report a multi-faceted study on several visualization solutions for displaying a collection of small objects discretely distributed on top of a dense visual representation. In particular, we have concentrated on the scenario of searching through a large-scale distributed file system with an enterprise search engine and visualizing the search results. Perceptually, discerning dots from a treemap is challenging due to many factors, including but not limited to dot size, change blindness, selective and divided attention [32, 168, 169, 195]. Hence, one may consider several alternatives to the static dots in Figure 5.1, e.g., using large dots or animating the dots. The approach of increasing dot size provides a further option for encoding more file attributes using *multivariate glyphs*. One possible benefit that a multivariate glyph may bring is the provision of an intermediate level of details between a dot and its detailed attributes, which are displayed in a popup window activated by hovering a pointer device over the dot. Viewing a coarse depiction of file attributes may facilitate fast elimination of incorrectly returned search results, and reduction of the needs for hovering over every dot (or glyph) to inspect its details. However, when using multivariate glyphs to depict search results as shown in Figure 5.2, the focus information may sometimes cause serious occlusion of the context information. Naturally, this leads to another design option of animating glyphs cyclically as shown in Figure 5.3. Such glyphs are referred to as *cyclically animated glyphs* (CAG).

Our multi-faceted study thereby examined three alternatives to static dots, namely (a) Cyclically-animated dots, (b) Static glyphs, and (c) Cyclically-animated glyphs.

Our theoretical rationales for considering cyclic animation and multivariate glyphs include:

- Animated motion, especially the onset of a motion, captures attention (e.g., [3, 164]). Motion pop-out effects have been observed in infants as young as 4.5 months old [56]. “Visual search for a moving target among stationary distractors is more efficient than searching for a stationary target among moving distractors” [180]. All these suggest that motion vs. static can provide focus and context information with different levels of attention stimulation.
- The human visual system and memory system allow us to capture multiple pieces of information displayed in an alternating manner at the same location. It is widely accepted that information about a visual stimulus remains in the sensory memory for up to a second [11], and in working memory between 15-30 seconds [162]. If attentively rehearsed and exercised (commonly in visualization), the retention of information in working memory can increase. If multiple data attributes are integrated into a visual representation (e.g., a multivariate glyph), working memory may hold more information [140]. This suggests that it is feasible to use cyclic animation to address the problem of occlusion as viewers can perceive both focus and context information, while the cyclic nature allows the visualization to remain effective as a tool of external memorization.
- Although animation may not always be effective in visualization (e.g., [83, 173]), the primary use of cyclic animation considered in this study is not for conveying information but for capturing attention, and resolving or reducing the problem of occlusion. Hence cyclically animated dots and glyphs may not suffer from the conventional drawbacks associated with some uses of animation in visualization, such as lack of external memorization capability, inaccuracy in motion perception and motion comparison, and poor pop-out effects in motion vs. motion comparison.

However, these theoretical rationales may not guarantee any of the three alternative approaches will offer effective visualization. In particular, glyph-based visualization inevitably adds additional burden upon users because they have to learn and memorize glyph encoding in order to gain much of the benefits from using multivariate glyphs. Cyclic animation can have many control parameters (e.g., timing and animation style), which may seriously affect the perception of the animated dots or glyphs. In general, the design space for both glyph encoding and animation control is huge, and for any single study, it is only feasible to explore

a small subdomain of such a design space. In this work, we conducted four empirical studies in addition to the necessary effort for developing a glyph encoding scheme and a prototype visualization-assisted search tool. Three empirical studies were carried out as controlled experiments, which allow us to discover perceptual effects of different *temporal transfer functions* for cyclic animation, and the just noticeable difference (JND) in the perception of some attributes encoded in glyphs. As controlled experiments can normally yield more precise findings when they are focused on one or a few control parameters, these enabled us to optimize two of the most important parameters for cyclically-animated glyphs with some degree of precision. Hopefully, the three alternative approaches can be compared with static dots on a reasonably fair basis. The fourth empirical study was conducted in the format of a focus group study, involving a short survey and an open discussion. The rationale for choosing this qualitative format is that it allows us to explore the possible merits and demerits of the three alternatives in many different aspects, which would be difficult for a few task-based control experiments to encompass.

These four empirical studies enabled us to gain several new insights. Firstly, all three alternatives can stimulate more attention in viewing search results overlaid on top of a treemap. Secondly, it is still feasible to perceive different attribute values even when a glyph is animated cyclically. Thirdly, participants preferred cyclically-animated dots and static glyphs, while most found cyclically-animated glyphs somehow irritating. Finally, participants expressed a fair level of willingness to learn and memorize glyph encoding, and many showed their capability of understanding a given encoding with little effort.

5.2 Related Work

Our work builds on research in the areas of treemaps, search results, and glyph-based visualization. We therefore briefly review the related work in these areas.

5.2.1 Search-related Visualization

Here we present focus and context and glyph-based techniques applied to information search result visualization. xFind [7] was developed to display search results as a ranked list as well as a scatter plot. Roberts *et al.* used glyph to visualize web search results [172]. Hoeber and Xue [105] developed HotMap and Concept Highlighter to show search results as two levels of detail: an overview of



Figure 5.1: The conventional depiction of search results as coloured-dots on a treemap results in great difficulties in visual search. In the background/context green are directories and purple are files.



Figure 5.2: When multivariate glyphs are used the occlusion would make the underlying context difficult to see.

the top 100 documents, and a detailed view of 20–25 documents at a time. They also discussed how these views support the visual query of web search results.

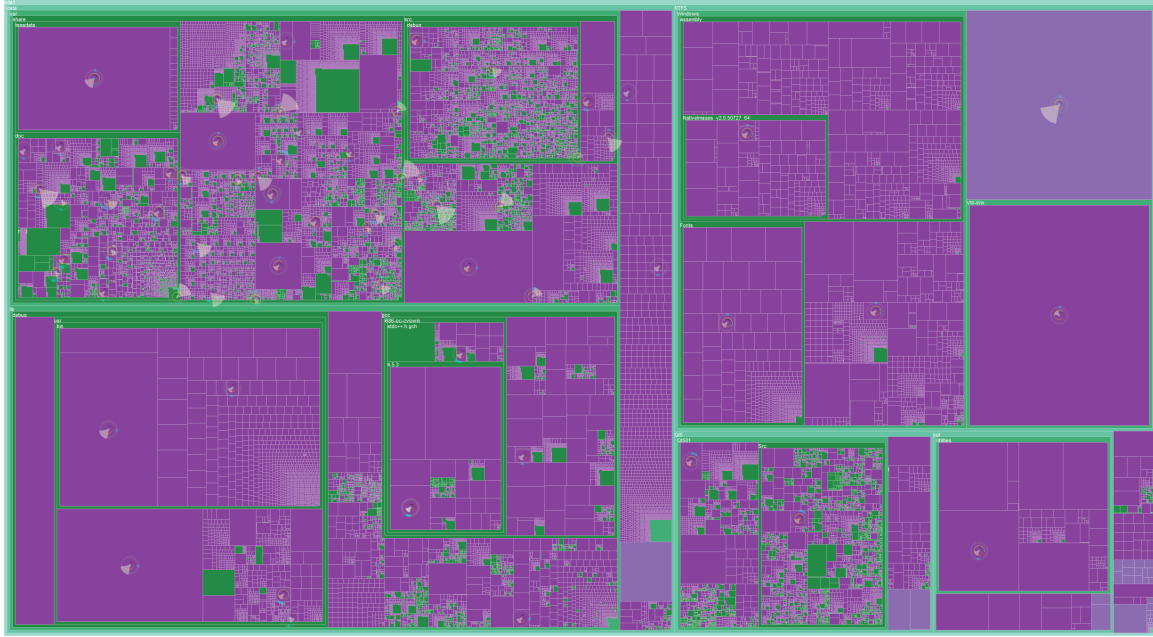


Figure 5.3: We propose to use cyclically animated glyphs to address both difficulties shown in Figure 5.1 and 5.2.

VisGets [66] and Fluid Views [67] were developed to visualize Web search results and the distribution of the results in a geographic map. Chau [42] proposed a glyph based on a flower metaphor to visualize web search results. Nocaj and Brandes [153] developed a tool to display the search results using a Voronoi treemap.

5.2.2 Treemaps

The concept of a treemap was proposed by Shneiderman [111], initially for visualizing the space utilization of a hard drive. Since then, the use of treemaps has been extended to almost any data with a hierarchical organization. Treemaps and pixel-based visualization share a similar dense appearance and pixel-based visualizations can also be regarded as a special case of a treemap. Keim provided a comprehensive survey of pixel-based visualization [118]. Oelke *et al.* proposed a number of methods for boosting some focus information in a pixel-based visualization [154]. Spatially ordered treemap was presented in [224] to show the geographic distribution of the hierarchical data.

5.2.3 Glyphs

Glyphs are visual entities composed of several visual channels representing multivariate attributes. Glyph designs tend to be largely influenced by particular application domains and tasks. Here, we mainly review the glyph-based visualization techniques relevant to information visualization. Ward [212] classified different glyph placing strategies, proposing rules for their usage and noted several bias issues in the context of information visualization. He proposed a perceptually-aware process of glyph generation via the mapping of data attributes to visual channels in [213]. Yang *et al.* meanwhile proposed Value and Relation (VaR) for visualizing multidimensional large datasets [227]. In information visualization, ring-shaped glyphs and radial mapping are common [104]. Further techniques on radial methods for information visualization were surveyed by Darper *et al.* [71]. A comprehensive review of glyph-based visualization for spatial multivariate medical data is proposed in [177]. Glyph-based visualization has also been applied in a wide range of domain specific data and tasks such as medicine [122, 146, 231], software [53], text [176], literature [1, 43], computation [221], scientific visualization [87, 104], geovisualization [18, 86, 115], sports [128], sensor data [13, 233], and biology [135, 145].

5.2.4 Animation in Visualization

The effectiveness of animation in visualization has been a debatable subject. In general, most agree that animation is an effective tool to aid learning and story telling (e.g., for explaining algorithms [205], or volume visualization [222]). Pritzkau *et al.* used animation for the visual boosting of scatter plots [165]. Farrugia and Quigley [80] used animation for visualizing temporal graphs. Robertson applied animation for presenting trends in multivariate data [173]. Ware and Bobrow [214] used animation in motion coding for patterns.

5.3 Design Considerations

Enterprise search engines do not guarantee consistent and accurate search results. For example, the search results often consist of *false positives* and *false negative*. In addition, enterprise search engines normally present the search results in a *list* or *tabular view*. These two shortcomings of enterprise search engines make it difficult

to (a) visually search for relevant files based on file attributes and (b) judge the quality of the search results. We propose visualization techniques to address these two difficulties (a) using glyphs to show multivariate information and (b) focus and context visualization to enable spatial awareness and thereby enable the identification of false positives and false negatives.

5.3.1 Multivariate Glyph

We considered many design options and finally selected a suitable design option based on a number of criteria such as (a) ability to encode the important file attributes, (b) discriminating capabilities when overlaying focus information on top of contextual information (see Section 5.3.2), (c) discriminating capacities of different visual channels, (d) metaphoric mappings for aiding learning and remembering, and (e) encoding costs in terms of pixel resolution.

A detail description of the file system glyphs can be found in Section 5.4. We considered four options (a) static dots, (b) cyclically-animated dots, (c) static glyphs, and (d) cyclically-animated glyphs. We conducted a focus group study to select a suitable option. A detailed description of the focus group study can be found in Section 5.7.

5.3.2 Focus and Context

“Focus+Context start from three premises: First, the user needs both overview (context) and detail information (focus) simultaneously. Second, information needed in the overview may be different from that needed in detail. Third, these two types of information can be combined within a single (dynamic) display, much as in human vision.” [38].

In order to display the search context (e.g., number of files, location of the files, file system structure, attributes, etc.) we considered two design options (a) node-link diagram and (b) treemap.

Using a node-link diagram, we can show some of the search space information, e.g., number of files, location of the files, and its structure explicitly. Thereafter, we can overlay the search results (as focus) on the node-link diagram using multivariate glyphs, showing more detail information about the search results. The main drawback of node-link diagram is that it does not utilise the display space

efficiently. In this work, we consider situations that involve large-scale distributed file systems, where files are stored across different locations on a network.

Treemaps can offer a cost-effective means of displaying the contextual information. Treemaps provide a compact display of directory hierarchy implicitly. It also allows additional visual encoding of one or two file attributes such as size and type. Because of the large number of files in such a file system, it is not feasible to depict every file in a treemap. We therefore expect a treemap to show only 3-5 levels of the hierarchy rather than showing the complete recursion. Most treemap elements will be folders rather than files in an overview. Hence a files may not be mapped directly to an individual treemap element. Most likely, it is displayed as an object on a treemap element, which is interpreted as a “file inside this parent, grandparent or ancestor folder”. We can overlay the search result glyphs (as focus) on the top of a treemap, showing more detail information about the search results.

5.3.3 Overview First and Details on Demand

In addition, the design implements the information seeking mantra: ‘Overview first, zoom and filter, then details-on-demand’ methodology proposed by Schneiderman [191]. For example, using the zoomable treemap a user can zoom into a specific region. The user can activate a tooltip window when the mouse pointer is hovering over an object, showing detailed information, see Figure 5.5.

This visualization technique typically result in dense information presented to the user. In dense visualizations, the user often cannot see the search results clearly, as illustrated in Figure 5.1. Here, we discuss the challenges of visualizing context information alongside large-scale multivariate data. However, existing techniques make it difficult to depict such multivariate focus information overlaid on top of a dense visual context such as a treemap.

5.3.4 Occlusion

Occlusion is a common problem in visualization. Occlusion occurs when two or more visual objects overlap, thus impeding the perception of one or more of these objects. As a result, users can misinterpret or fail to perceive the important details of a visual element. Occlusion becomes a major problem in large-scale data visualization. As the density of data increases, the chances of overlapping two or more visual objects rapidly grows. In Figure 5.1, we highlight the focus over a

treemap context by assigning a primary glyph (dot) to each data point of interest (the search result). A primary glyph could also display the multivariate information. As the entire search space is large, each primary glyph representing the point of interest will shrink to a small glyph. The main problem here is the occlusion between the primary glyph and the background treemap. The occlusion makes it difficult to perceive several portions of the treemap as well as primary glyph. In large-scale information visualization, accuracy and response time are common problems. When visual objects overlap with each other, it is difficult to perceive the number of objects and their visual properties such as size, shape, colour, etc. This can lead to a misinterpretation of data. The visibility of the primary glyph could be improved by adding a secondary glyph, as shown in Figure 5.2. The secondary glyph could also encode several data attributes. However, as visual objects overlap with each other, it is difficult to perceive the quantity, colour or shape of numerous secondary glyphs.

The occlusion problem can be alleviated by using semi-transparent lines or by colouring the boundaries. The glyph placement strategy is also crucial in achieving meaningful and enhanced glyph-based visualizations. However, these modifications may still fail when the dataset is large and focus and context are to be visualized together. Glyphs that occlude each other or occlude large portions of the treemap can easily counteract the information increase intended by their use. Therefore, to minimize the glyph occlusion as well as to achieve a meaningful and enhanced glyph-based visualization, we propose *cyclically animated* glyphs, as shown in Figure 5.3.

5.4 Visual Mapping of Glyph

The glyphs in our system are composed of several visual channels presenting the multivariate attributes associated with the EXT4 file system data. In an EXT4 file system, the *stat* structure contains the *inode* information and stores 13 different attributes related to a file. Encoding these 13 attributes in a single glyph posed a significant challenge, mainly because (a) it requires a high pixel resolution to encode all attributes; (b) visual channels would lose their discriminating capacity under low-resolution conditions; (c) the visual context is dense, and (d) the time attribute (*time_t*) contains multiple components: time and calendar date; hence, two different visual channels are required to encode each time attribute.

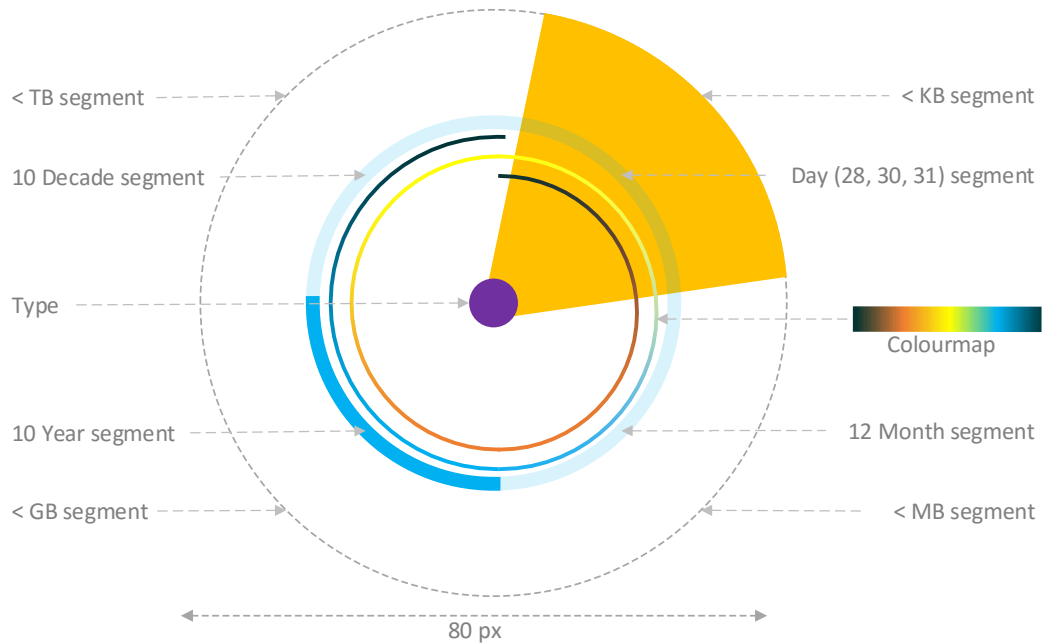


Figure 5.4: File attribute glyph. Four different parameters (type, size, date and time) are encoded into the glyph.

We began our design activities with a series of brainstorming meetings. In these meetings, we identified the most commonly-used file attributes: type, size, time, date and protection mode. We produced a table consisting of several optional visual channels for each file attribute. After several iterations, a suitable option for each file attribute was identified on the basis of occlusion, clarity, visual separation, metaphoric abstraction for aiding learning and remembering, and encoding costs in terms of scalability. A schematic diagram of our final glyph design is shown in Figure 5.4.

File type. Both colour hue and shape are effective visual channels for categorical attributes. We considered shape as a visual channel for mapping the file types. However, there is a strong interdependence between shape and size. When the available display size is small, the number of discriminable variations for shapes becomes small. File types are thus represented as saturated colours on a five-pixel dot. We adapted the file categorization and colour-coding scheme of GNU/Linux, and divided files into n categories, each is assigned to a light colour (assuming a dark background). By default, we have 8 categories of files. Both the categorization and colour mapping can be modified by users.

Size. The file size is a numerical attribute. We considered several design options, including the area of a shape, and length of an arch. However, a file system consists

of diverse varieties of different sized files. The same visual channel cannot support discriminable variations at the gigabyte level as well as the kilobyte level. We thus divided a value of file size into four components, 0-999 bytes, 0-999 kB, 0-999 MB, 0-999 GB. Each component is shown as a pale yellow pie segment at one of the four quadrants of the glyph. This enables users to focus selectively on a specific quadrant, for instance, in searching for a specific range of sizes, or for comparing file sizes indicated by different glyphs. For file size of a TB or above, we replace each yellow segment with a red segment, such that TB in place of bytes, PB in place of KB, EB in place of MB and ZB in place of GB. The radius of the pie segment is 40 pixels and the spanning angle is 70° .

Time. The clock metaphor is a natural choice for encoding the time attribute. We explored several designs of the metaphor based on a pointer moving around a circular object. In order to cover the range of [00:00:00, 23:59:59], we used a spiral to facilitate the 720° rotation of the pointer. Because it is not intuitive to determine which half of the range the inner (or outer) circular segment represents, we added a metaphoric colour map. As shown is Figure 5.4, the continuous colourmap features black, orange, yellow and cyan at the 0, 6, 12, 18 hour positions, symbolizing night, morning, noon and afternoon respectively. The combination of spiral location and colour symbolism provides a dual visual encoding of time for fault tolerance in perception. The largest and smallest diameter of the spiral is 38 and 30 pixels respectively.

Date. The date attribute in a file system is normally displayed in text using one of the international or regional formats (e.g., YYYY-MM-DD). In the context of file searching, it is more useful to encode the age of a file, i.e., the interval between the date of searching and the record date of creation (or last modified/accessed). The main challenge in visually encoding the date attribute is the large range of values, and different search tasks usually focus on a specific range. We thus adopted a design concept similar to that for **size**. We divided file ages into four categories: day, month, year and decade. A translucent grey ring of 40 pixels in diameter is divided in four quadrants. The first quadrant represents the range of 0-31 days, the second 0-12 months, the third 0-10 years, and the forth quadrant 0-10 decades. A non-zero value in each quadrant is encoded as the length (or degree) of a coloured circular arc, extending clockwise. We considered the option to display creation, modification and access time/date all together simultaneously. However, we could not find a suitable mechanism to separate these six attributes visually. Therefore, we decided to show one attribute for date and one for time, with an option for users

to choose different attributes.

Protection mode. The protection mode attribute is composed of nine access permissions for [read, write and execute] $\times$ [user, group and others]. We tried out a number of design options. They all resulted in noticeable visual clutter. We thus decided not to consider glyph encoding of the protection mode in this chapter, though we have implemented one design in our system as an option requiring explicit activation by the user.

In our user interface, glyphs are displayed on top of a squarified treemap [33] showing the search space. The default search space is set according to the access status of a user, e.g., the entire file system for a superuser. The treemap displays the top k -levels of the search space. As this chapter focus on large-scale distributed file systems, we assume that not all files or directories in the search space can be represented by the treemap. For example, our testing environment (simulated data) consists of more than 500,000 files. We thus use the colours dark green and purple to represent directories and files respectively. We indicate different directory levels through the use of different shades of green and purple, with lighter shades indicating higher levels, and darker shades indicating lower levels. The default depth k is set to 3.

Once the user enters their search criteria, the results of the search are shown as glyphs overlaid on top of the treemap, as shown in Figure 5.5 (a). This figure shows more than 200 search results on top of a treemap with five levels. The zoomed view of a region shows around 17 results associated to the region zoomed here, shown in 5.5 (b). As this study also includes static dots, cyclically-animated dots and cyclically-animated glyphs, these are all options available through a *visual mapping dialog*. The dialog window is used mainly to define the mapping of file attributes to visual channels of the glyphs. For cyclic animation, there are additional options for choosing the animation style and adjusting the animation tempo, which will be discussed in the next section.

5.5 Cyclically Animated Glyph (CAG)

One potential drawback of static glyphs is the occlusion of the dense contextual information in the treemap, such as the density, partition and colour shade of treemap cells. One objective of this chapter is to examine if cyclically animated

5. SEARCH RESULT VISUALIZATION



(a) Focus (glyphs of files) and context (treemap) view shows the distribution of the search results in search space. Now, we zoom into the region Canada (top-right).



(b) The zoomed view shows the zoomed area (Canada) and the results (using bigger glyphs). A tooltip window displays the multivariate information in text that is activated by hovering the glyphs. Green and purple colour represent directories and files respectively. Different saturation of green is used to highlight filesystem depth.

Figure 5.5: Visualization of file search, featuring glyphs for search results (focus) and a treemap for the file system (context).

glyphs can alleviate this problem by allowing context and focus to share a temporal visual channel. The introduction of cyclic animation may lead to issues that are not associated with static glyphs. These issues include *animation style* and *tempo control*. The latter is defined by a *temporal transfer function*. It is important to note that although the human visual system is finely tuned for detecting peripheral and non-peripheral motion as well as motion signatures (e.g., motion frequency and direction) [228], the animation in our study is solely for alternating focal and contextual information and for attracting attention. The temporal patterns of the animation do not encode additional information.

Animation Style. We considered several animation styles, namely changing size, colour, and opacity. Because colour is a highly attentive visual channel while the perception of colour encoding is easily confused, we ruled out colour-based animation at an earlier stage of the design. Both size and opacity based animation have their relative merits and demerits. In addition, whereas a single animation style can be applied to a glyph, one can also apply different animation styles to different visual channels of a glyph.

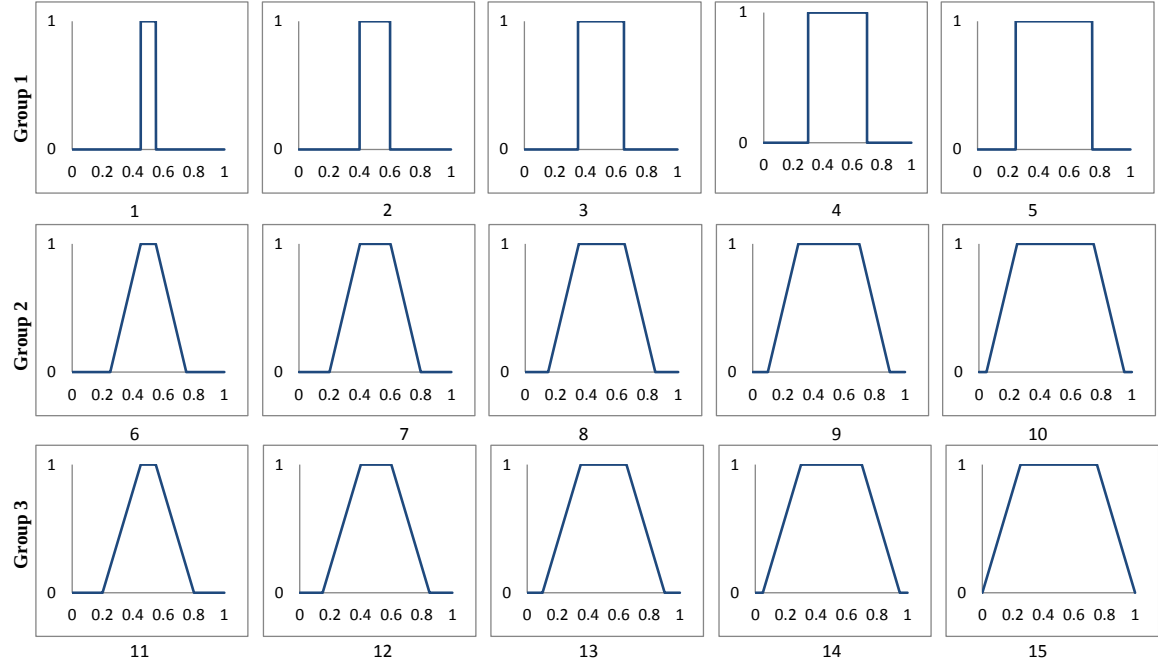


Figure 5.6: Design options of temporal transfer functions. Each TTF is shown as a time series, with x -axis shows the normalized time, and y -axis shows the switch between focus (1) and context (0). The five TTFs in each row represent different focus/context ratio, while the three rows represent different transition periods from minimum to maximum. The values of transition periods are 0, 300, and 375 ms.

Temporal Transfer Functions. In cyclic animation, temporal transfer functions (TTF) are used to control the tempo of the changes of a visual channel over time. The TTF should be carefully designed so that users can perceive both the focus (i.e., glyphs) and the context (i.e., the treemap) comfortably. Each TTF defines the overall cycle length of the animation, which is divided into four subsections: (a) context only, (b) transition from context to focus, (c) focus only, and (d) transition from focus to context. Smooth transitions and an adequate repeat rate are obvious criteria for an acceptable animation. If the overall length or any subsection length is too long, the user may be impatient about waiting for the switch between the focus and the context. If the cyclic frequency is too high, the user may not have sufficient time to observe the visually-encoded information in either the focus or the context [85]. Furthermore, if the transition is too abrupt, the glyphs may irritate the user. If the transition takes too long, it is wasteful in using the temporal bandwidth as the visual information during the transition is least useful. Figure 5.6 shows various design options for the TTF, which were identified in a pilot study. We realized that discovering an optimized TTF is a non-trivial task. It would vary from one person to another, and from one visual channel to another. Nevertheless, it would be better to collect some quantitative data to guide the design of TTF in addition to the qualitative judgement of the research team.

5.6 Pre-development Empirical Studies

In this section we report three empirical studies that were designed to investigate the effects of temporal transfer functions on the perception of cyclically-animated glyphs (CAG), and to collect quantitative experimental data to guide the design and implementation of CAG. With relatively optimized settings for CAG, it would make the comparison with other design options (i.e., static dots, cyclically-animated dots, and static glyphs) reasonably fair. The comparison will be reported in Section 5.7.

The three pre-development studies were conducted as controlled experiments. *Experiment One* examined the effects of different TTFs on the perception of contextual information occluded by a glyph. *Experiment Two* examined the effects of different TTFs on the perception of focal information when the glyph is animated. *Experiment Three* examined the just noticeable difference (JND) of a visual channel when the glyph is animated. Experiments One and Two were designed mainly to

help find the most effective TTF whereas Experiment Three was designed mainly to discover the JND.

Participants were presented with the experiments in the same order, i.e., Experiments One, Two, Three. All experiments used a “2 Alternate Forced Choice” (2AFC) design in which participants were required to select one option from two possible options presented on screen. All three experiments used randomly-ordered simple stimuli to minimize confounding effects. Example stimuli of these three experiments are presented in Appendix B.

5.6.1 Controlled Experiment One

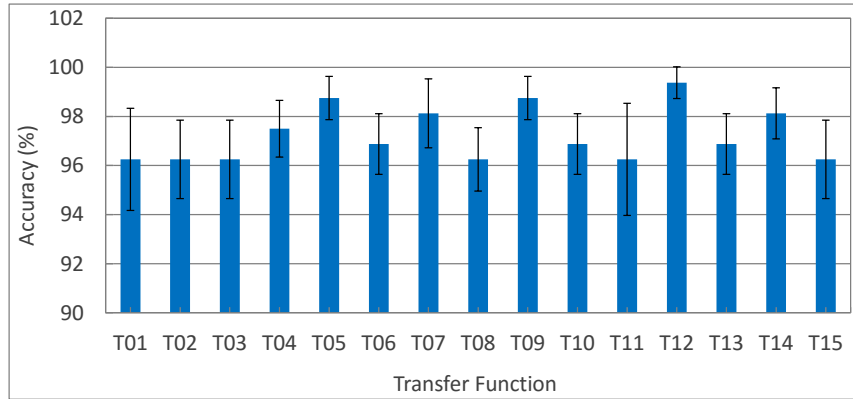
This experiment examines the effects of different TTFs on the perception of contextual information. Based on the file search scenario, the stimuli of the experiment shows a small treemap, and its central region is occluded by a CAG. Each stimulus is shown with one of the 15 TTFs in Figure 5.6. Given that the glyph will cyclically animate until a selection is made, it was anticipated that the TTFs of the CAG will have no reliable effect on the accuracy. However, an effect of TTF on response time was anticipated.

Participants. 16 participants (14 male) took part in this experiment in return for a £10 voucher. Participants were recruited from the postgraduate student and staff communities. All participants had normal or corrected to normal vision and all were naive as to the aims of the study.

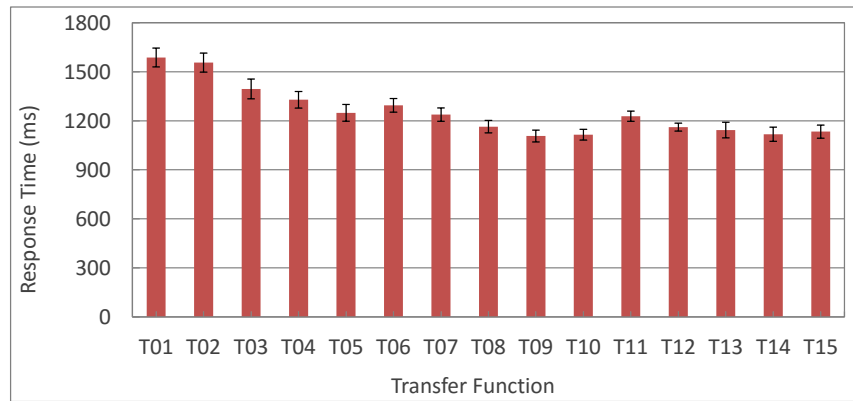
Apparatus. The experiment was conducted in a well-lit computer lab, lit uniformly by overhead fluorescent lighting. Stimuli were presented on a Dell Ultrasharp 24 inch IPS widescreen monitor at 1920 x 1200 resolution and 32bit colour depth, using Fedora GNU/Linux 17. Computers had an Intel i-5 3.3GHz Processor, and 4GB RAM. All three experiments were presented to participants using the same computer.

Stimuli. Stimuli consisted of a simple treemap (400 x 400 pixels), comprising a white background and black border lines, see Figure B.1. At the center of the treemap was a smaller region (60x60 pixels). Superimposed over this region was a CAG with an opacity-based animation controlled by different TTFs. Below this treemap were two smaller images (60x60 pixels), one of which (randomly drawn from 20 different patterns shown in Figure B.2) matched the occluded region at the center of the treemap.

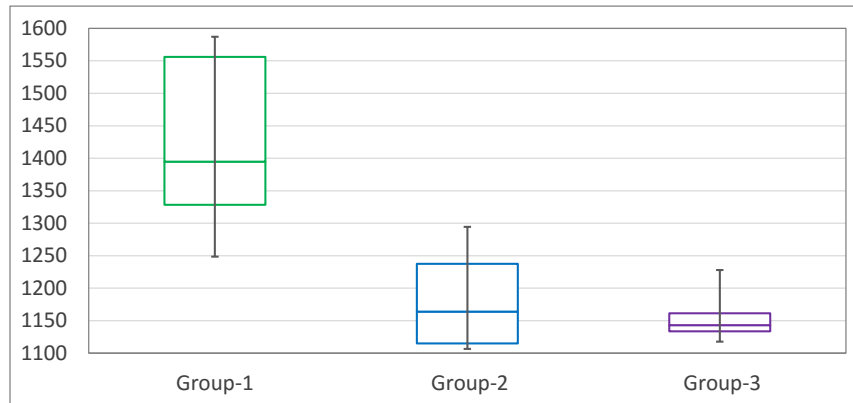
5. SEARCH RESULT VISUALIZATION



(a) Accuracy



(b) Response Time



(c) Response Time By TF Group

Figure 5.7: Results of the Controlled Experiment One.

Procedure. Participants completed a total of 150 trials, with which each of the 15 TTFs were randomly presented 10 times. Once participants had made their choice, there was an inter-stimulus interval of 250ms, during which time the screen was filled with white noise. Following this the next stimulus was presented. This sequence was repeated for all trials. After every 50 trials, a message was presented

advising participants to take a short break.

Results. Accuracy data (Figure 5.7) shows all participants achieved 96% or above, with very little differences between the TTFs. A one-way repeated-measures ANOVA confirms this ($F_{(14,210)} = .741, p = .731$). Response Time data (Figure 5.7) shows clear differences between TTFs, confirmed by a one-way repeated-measures ANOVA ($F_{(14,210)} = 34.274, p < .001$).

Discussion. Observations from this experiment confirm our hypothesis that there will be no effect on accuracy of the CAG due to occlusion. Further, the response time data demonstrates the advantages of the use of well-designed TTFs, as participants showed faster response times for TTFs with smooth transition between focus and context (e.g., TTF 6-15) TTFs with abrupt transition (TTF 1-5) resulted in slower response time.

5.6.2 Controlled Experiment Two

This second experiment examines the effects of TTF on the perception of focal information encoded in a CAG. We selected *size* as the visual channel to study as different channels would require separate experiments. The participants and apparatus are the same as in Experiment One.

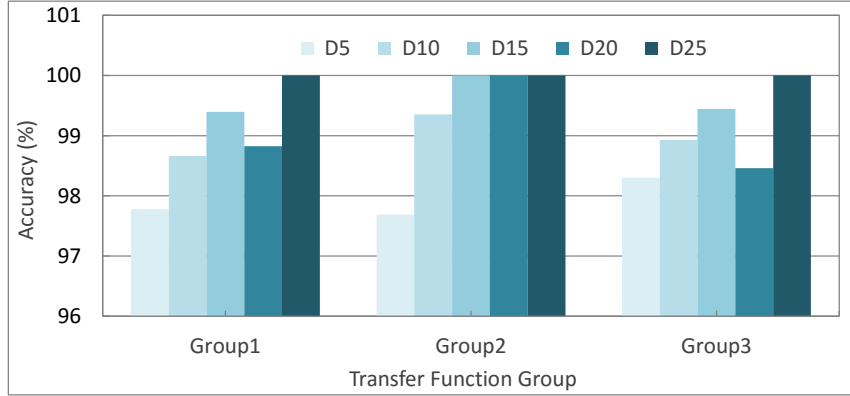
Stimuli. A total of 150 stimuli were used for this experiment. Each stimulus is comprised of two white boxes with thin black borders. At the center of each box was a CAG, see Figure B.3. The glyphs and TTFs used were the same as those used in Experiment One.

Procedure. Participants were asked to judge which of the two glyphs on screen was larger. Selection was made using the ‘D’ key for the left image, and the ‘K’ key for the right image.

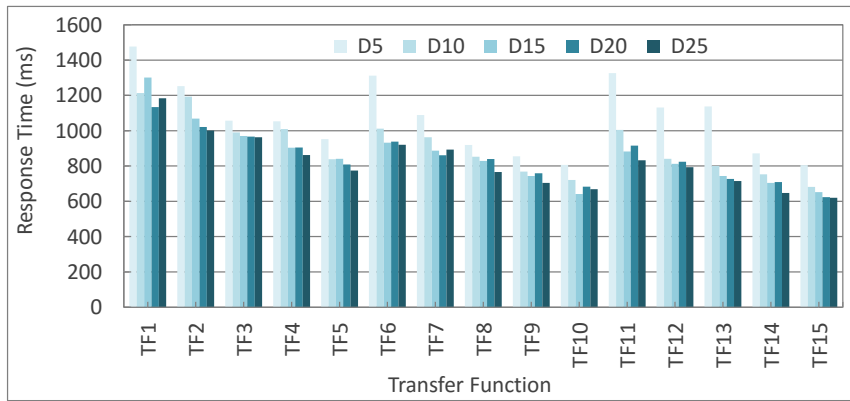
In this experiment, we used six different stimuli of size 15, 20, 25, 30, 35, and 40 pixel (see Figure B.4), such that the size difference between each glyph was systematically varied to be 5, 10, 15, 20, and 25 pixels, with these differences being randomly assigned. Each trial was followed by an inter-stimulus interval of 250ms during which time the screen was filled with white noise. As with Experiment One, 15 TF’s were tested, with each TF being tested 10 times. This led to a total of 150 trials. After every 50 trials, participants were advised to take a short break.

Results. Accuracy data (Figure 5.8) shows the expected reliable differences as a function of size difference, with accuracy decreasing as the size difference decreased ($F_{(4,60)} = 3.222, p = .018$). Consistent with Experiment One, data shows

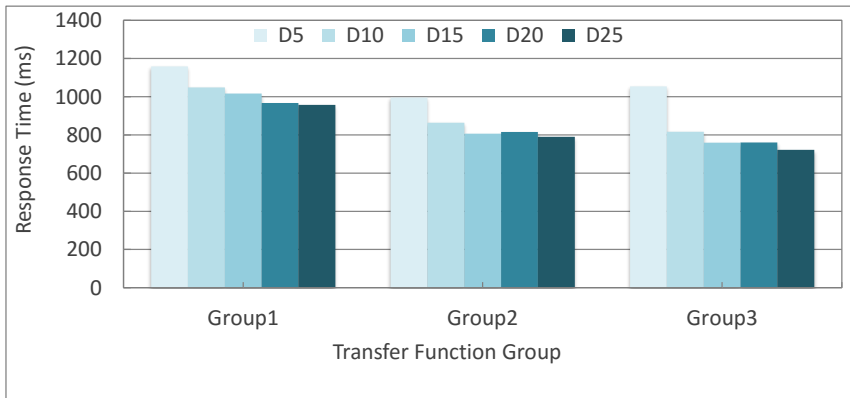
5. SEARCH RESULT VISUALIZATION



(a) Accuracy By TF Group



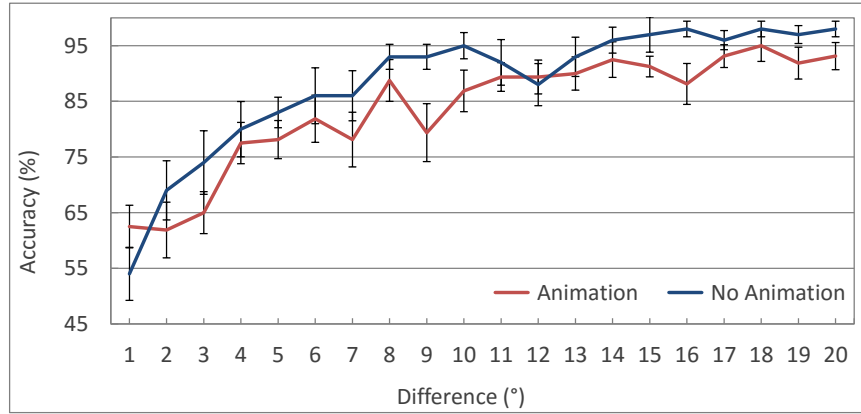
(b) Response Time



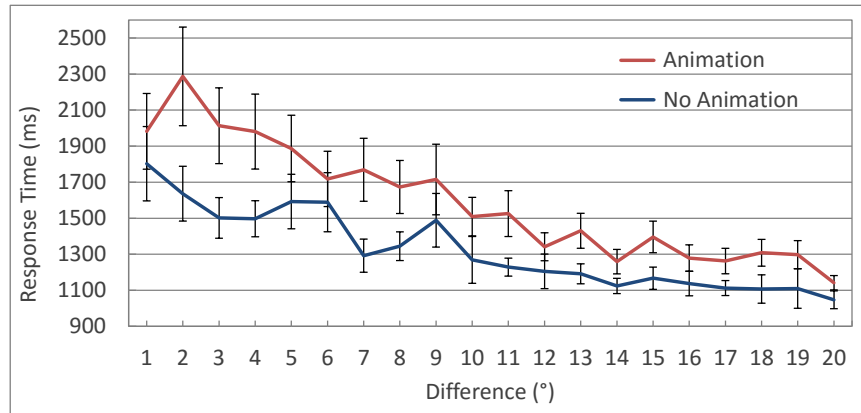
(c) Response Time By TF Group

Figure 5.8: Results of the Controlled Experiment Two (D5: 5 pixels difference, D15: 15 pixels difference, and so on).

no reliable differences in accuracy between the TTFs. Response time data (Figure 5.8) shows the anticipated decreases as a function of size difference, with larger size differences recording decreased time ($F_{(4,60)} = 17.929, p < .001$). More importantly, the response time data shows reliable differences between TTFs groups



(a) Accuracy



(b) Response Time

Figure 5.9: Results of the Controlled Experiment Three.

($F_{(14,210)} = 41.022, p < .001$). Figure 5.8 shows the response time data of the three TTF groups, i.e., 1-5, 6-10, and 11-15 (Figure 5.6). The data for each group is further divided into sub-groups based on the size differences between two glyphs in each stimulus.

Discussion. Experiment Two has shown no effect on focal information in terms of accuracy when using CAG. As Experiment One has shown no effects on contextual information in terms of accuracy, these findings confirm that cyclic animation can facilitate the observation of information in focus as well as occluded contextual information. Similar to the finding in Experiment One, TTFs with smooth transition resulted in faster response time in most cases, except it is less conclusive when the size difference is small.

5.6.3 Controlled Experiment Three

This third experiment was designed to establish the just noticeable difference (JND) when a glyph is animated. We selected *arclength* as the visual channel to study as different channels would require separate experiments. Initially a minimum difference of 2° , and a maximum difference of 30° with 2° steps was planned. After a pilot study showed little difference in accuracy above a difference of 15° , the minimum difference was changed to 1° , and maximum to 20° , with increments of 1° . The apparatus used for this study was the same as that used in the previous two experiments.

Stimuli. A total of 200 stimuli were used for this study. Each stimulus contains two rectangular white boxes with thin black borders. At the center of each box was a glyph, comprising a grey circle (diameter 60 pixels) made of a thin line (5 pixels width), see Figure B.5. A coloured arc of 5 pixels in width would move around the circumference of a grey circle from zero to a maximum arc length before returning to the start position. All animated stimuli used TTF No. 15 in Figure 5.6.

At first a stimuli with rotation 10° to 350° (Figure B.6 (a)) is randomly generated and randomly placed in the left or right window. We avoid 90° , 180° , 270° , and 360° to avoid confounding effects. A delta value between 1° - 20° (Figure B.6 (b)) is randomly chosen. A second stimuli subtracted the delta from the first and was placed on the other window.

Procedure. Participants were tasked to identify which line moved farthest around the circumference of the circle. The difference between the two distance was systematically varied from 1° to 20° in 1° increments. Participants made their selection using the 'D' key for the left-hand image, and the 'K' key for the right-hand image.

Results. Accuracy data (Figure 5.9 (a)) shows an increase in recorded accuracy as size difference ($^\circ$) increases. A one-way repeated-measures ANOVA confirms this, reporting a main effect of size difference on accuracy ($F_{(19,285)} = 12.042, p < .001$). Response time data (Figure 5.9 (b)) shows corresponding decreases as size difference increases. A one-way repeated-measures ANOVA confirms a main effect of size difference on response time ($F_{(19,285)} = 12.342, p < .001$).

Discussion. For the JND analysis, Figure 5.9 indicates one linear band in the line plot for the animated arc channel at 4° - 20° . An estimation of 75%-JND for the animated arch channel is about 4° .

5.6.4 General Discussion

Overall, we can make a number of observations from these three experiments. (a) Using different TTFs has limited effects on the accuracy of the participants' performance. (b) Using TTFs with different transition settings has noticeable effects on the response time of the participants' performance. (c) Smooth transitions in TTFs is more advantageous than abrupt transitions. (d) The estimated JND values for animated arc channels offer a useful guideline in using such channels in glyph design. It is highly desirable to collect more JND values for other visual channels. In general, the three pre-development experiments have offered useful quantitative data for identifying a good temporal transfer function to be implemented in the system, and offered a reasonably fair basis for the post-development evaluation.

5.7 Post-development Evaluation

5.7.1 Organization

After implementing a file search interface with four different design options, namely (a) static dots, (b) cyclically-animated dots, (c) static glyphs, and (d) cyclically-animated glyphs, we conducted a further empirical study to compare these four design options. Although such a study may theoretically be carried out in the form of controlled experiments, the range of factors that may affect the user performance is rather extensive. As there has not been any previous study on this topic, the first step of evaluation was in fact to establish a list of such factors. We thus chose one of the commonly-used qualitative research methodologies, *focus group discussions*. The participants were 10 individuals from the staff and postgraduate student communities. All had experience of using search facilities currently available in various file systems. We organized 10 participants into two equal-size group meetings with the same schedule plan to achieve a proper balance between collective views from the group and strong voices within the group. It also gave a good balance between learning visualization techniques and offering opinions.

Introduction (20 minutes). Each meeting starts with a 20 introductory presentation. Participants were informed of the overall goal of the system, application scenarios, and objectives of the meeting. They were then introduced to the concepts of treemap and glyph-based visualization. They were shown four design options, each is accompanied by a relatively comprehensive list of possible pros

and cons compiled by the research team. The word “possible” was emphasized to avoid leading the participants. By giving a balanced list of possible pros and cons, participants were made to feel that they were being consulted rather than being tested. **Survey** (20 minutes). Following the introduction, participants were asked to complete a questionnaire with 14 questions, see Section B.2. This allowed us to collect views from every individual participant in an equal manner, and to compile them into a coherent set of collective views. The 14 questions, which were given in the supplementary materials, were divided into four groups, namely (a) static dots vs. animated dots, (b) static dots vs. static glyphs, (c) static glyphs vs. animated glyphs, and (d) animated dots vs. animated glyphs. When answering the questionnaire, example visualizations were shown side-by-side using two projectors.

Open Discussions (50 minutes). Following the questionnaire session was an open discussion session, during which participants were encouraged to offer feedback and comments relating to design issues, especially those not covered by the questionnaire. In both meetings, the discussions were lively, and some participants were highly enthusiastic in offering their opinions and suggestions.

5.7.2 Survey Results

Static dots vs. animated dots. The results of the static dots vs. animated dots are presented in Figure B.7. Results show a strong preference towards animated dots. 80% of participants expressing such a preference, only 10% of participants preferring static dots, and 10% no clear preference. 40% of participants rated animated dots as either mildly annoying, or annoying. This indicated that cyclic animation can provide additional attentive power, which is necessary for identifying small objects distributed around a treemap.

Static dots vs. static glyphs. The results of the static dots vs. static glyphs are presented in Figure B.8. The results indicate a willingness to learn the visual encoding of the glyph (90%), even though many found such learning and remembering to be difficult (70% and 60% respectively). Many (60%) agreed that glyphs will be advantageous when one needs to perform 20 or more searches per day.

Static glyphs vs. animated glyphs. The results of the static glyphs vs. animated glyphs are presented in Figure B.9. Results show a general preference for static glyphs. Most found it easier to identify individual search results (60%), and see individual file attributes (70%). Most rated animated glyphs as either mildly annoying or annoying (80%).

Animated dots vs. animated glyphs. The results of the animated dots vs. animated glyphs are presented in Figure B.10. As expected, there was little difference when asked which afforded better views of the underlying directory structure, with 50% and 40% rating animated dots and animated glyphs respectively as giving the best view.

5.7.3 Summary of Open Discussions

The open discussion session (about 20 minutes) was recorded in order to transcript their views more accurately. No written statement was taken, but the participants were encouraged to write down any additional comments and suggestions. In the general discussions, we were interested in their comments and suggestions about:

- (a) The design of glyphs, including but not limited to visual encoding for different attributes, overall glyph size, use of colours, and so on.
- (b) The timing of the animation for dots or glyphs.
- (c) If there is irritation or annoyance with animated dots or glyphs, how can it be alleviated?
- (d) Mechanisms for leaning and remembering glyph encoding.
- (e) Any other important aspects.

During the discussions, participants asked several questions on the visual encoding of the glyph and showed much interest to learn. Many participants learned the visual encoding very quickly, and made informed suggestions about various design aspects. For example, one participant pointed out that the visual encoding of the size attribute should support file sizes of a terabyte or more. We accommodated this suggestion in a revised design. Some participants pointed out an important factor that even without any animation a glyph is easier to see than a dot, due to the size factor. Naturally, there were some suggestions about our use of colour. A few suggestions led to debate among participants.

The participants made a number of positive comments. In particular, visualization can be useful to quickly interpret the search results and their location in a file system instead of reading any detailed text. Some expressed that in a traditional file system search, when the search results were displayed then they had to go through each result one by one. Sometime locating one result required navigating

through many levels of hierarchy. Repeatedly performing such tasks was a very tedious job. The new visualization could eliminate this problem, and therefore could be very useful.

A fair amount of discussion was focused on the concept of treemaps as the background context, with some participants finding it difficult to appreciate the usefulness of a recursive directory-based cell layout versus their own alternative suggestions such as a calendar layout. The discussion also confirmed that animated dots were helpful as static dots were difficult to see, whilst views on the animated glyphs remained negative. Many participants considered that even as animation frequency was slowed, animation could still prove to be distracting. Some participants suggested providing options for switching on and off individual visual channels as reading and comparing a specific attribute could be easier.

5.8 Conclusion

We have conducted a study on glyph-based visualization over dense visual context. We used file search in a large file system as the scenario to facilitate the exploration of various design options as well to provide a frame for the four empirical studies. The three controlled experiments have provided the first set of quantitative data for optimizing temporal transfer functions. They have also confirmed the fact that participants can perceive focal information and occluded contextual information with an appropriate temporal transfer function.

Our empirical work showed a number of important findings relating to the use of CAG. Most importantly, we showed in Experiment One that the use of animated glyphs as focus had no negative effect on performance in terms of accuracy for context information. Similarly, Experiment Two showed no negative effect on performance in terms of accuracy for focus information.

Our consultation with the potential users in a focus group study has indicated the following: (a) Users are generally willing and able to learn glyph encodings if the task of file search needs to be performed frequently in a large file system. (b) Users generally prefer cyclically-animated dots and static glyphs in comparison with static dots and cyclically-animated glyphs. (c) Potential users are in agreement that static dots are ineffective in supporting visual search due to their limited visibility, and (d) Most of users may find cyclically-animated glyphs irritating in a visual interface, though they can be a workable option for some.

The next chapter focuses on the visualization of provenance information related to the searches collaboratively performed by administrators.

6

Search Provenance Graph and its Visualization

Contents

6.1	Introduction	119
6.2	Related Work	121
6.3	Search Provenance Graph (SPG)	123
6.4	Computing Query Similarity for SPG	126
6.5	Visualization of SPG	129
6.6	Evaluation	138
6.7	Conclusions	142

6.1 Introduction

File search is an explorative process, during which a user progressively refines search criteria by observing and learning from successes and errors in the earlier search results. Visualization-assisted search tools have been found to be effective in supporting search activities, e.g., assisting in iteratively reformulating queries and visualizing search results (e.g., in Web search [66]). The quality or validity of data can be examined or reviewed in the context of its original source and transformations over time [37]. The historical information that need to be captured over time to support that process is called “Provenance Information”. Search and exploration processes can further be supported by using provenance visualization [36].

This work was conducted in the context of an industrial application, where intensive file search is part of an industrial process and is to be performed by a team on a daily basis. As the total number of files related to projects could easily be in tens or hundreds of thousands, and many are distributed among different teams and stakeholders in the company, the need for file search in a large distributed file system becomes inevitable. As part of a feasibility study led by a large construction enterprise, we developed a prototype system that provides visualization-assisted search capabilities. The objective is to enable the enterprise to assess the merits of visualization in comparison with two other approaches, database without visualization and search engine without visualization.

In Chapter 4, we presented an enterprise search engine, enabling the experimentation with visualization facilities. Thereafter, in Chapter 5 we presented a glyph-based visual interface for visualizing search results in conjunction with a treemap. Following this early development, in this chapter we focus on developing a *Search Provenance Visualization* capability that allows users to explore search criteria (i.e., their hypotheses), and to reformulate search criteria progressively. It is necessary to note that the goal of a user's task is not to formulate the best set of search criteria, but to work efficiently in collaboration to find all required files. Once all the required files have been found, the user will likely stop the process even if the best set of search criteria has not been found. In many cases, the best strategy may be to find a complete collection of the required files using a small number of queries even though the search criteria for each query may not be optimal. Meanwhile users' experience of previous projects will play a significant role in formulating search criteria and an effective search strategy. When a search process takes a few days and/or involves more than one administrator, visualization support for search provenance is essential.

The main contributions of this chapter are:

- We have introduced a graph structure called *Search Provenance Graph (SPG)* for recording the search provenance in individual projects and for enabling provenance visualization.
- We have proposed an algorithm for computing semantic similarity (using ontologies) in constructing and updating SPGs.
- We have developed combined glyph-graph visual representations for visualizing SPGs.

- We have worked closely with a management team in the industrial partner to compare three approaches for supporting enterprise search operations, and assisted the industrial partner in making a strategic plan for meeting the new BIM requirements to be implemented in 2016 [206]. Our visualization-assisted approach compared favorably with alternative approaches under investigation.

6.2 Related Work

Our work builds mainly on two areas of research (a) *Information Retrieval (IR)*: search strategies, query reformulation, and query similarity measurement, and (b) *Visualization*: visualization-assisted search, provenance visualization, ontology-assisted visualization, and glyph-based visualization.

6.2.1 Search-related Visualization

Many research papers have been published by the visualization and IR communities on navigation and exploration of the Web information space.

VisGets [66] was developed to visualize different Web search attributes, to assist in formulating search queries and to visualize search results using interactive geographic maps and tag clouds. Applications of VisGets to a variety of Web data collections were demonstrated in [69]. An empirical study on interactive and visual exploration of the Web using VisGets was reported in [70]. Fluid Views [67] allows users to view Web search results geographically and temporally in dual layers. PivotPaths [68] allows users to explore datasets with a variety of relation types across various dimensions. A Voronoi treemap to organize search results was proposed in [153]. ProjSnippet [91] visualizes web search results in a global view and provides a clustering technique for identifying similar content. The similarity measurement used in this work is based on the vector space model. Footprints [109] was developed for analysts to help them search documents and articles.

6.2.2 Provenance Data Visualization

Ragan *et al.* reviewed the provenance information-related articles published in both visualization and data science literature in [166]. They discussed different types of provenance information (e.g., data, visualization, interaction, insight, and

rationale) and its use cases (e.g., for recall, replication, action recovery, collaboration, presentation etc.). This review provided a systematic method for organizing and comparing systems supporting provenance, and their design evaluations. Kreuzeler *et al.* [124] proposed a tree structure to represent change histories of an exploratory data mining application. They also described a mechanism to calculate undo and redo operations in this tree structure.

The visualization technique commonly used for provenance data is the node-link diagram [37]. For example, such diagrams are used for visualization workflows in VisTrails [16, 36, 184], scientific workflows in [23], and biological experiment workflows in [134, 135]. They are also used in provenance management tools such as Haystack [107], Probe-It [58], and Orbiter [133]. These node-link diagrams feature a variety of visual encoding methods, such as including directed graphs [58, 107], collapsible summary nodes [133], and glyph representations [134, 135]. Dunne *et al.* proposed visualization of analytic provenance in [75]. Borkin *et al.* [29] proposed the InProv tool which employ a radial layout with hierarchical encoding of the time dimension. Walker *et al.* proposed a XML-based provenance model to support data exploration and to visualize provenance in the context of Human Terrain Analysis in [209].

6.2.3 Search Provenance or Log Visualization.

Web search activities generate query logs [229], usability logs [92], and web click-streams [219]. The SDSS Log Viewer [229] was developed for identifying users' data-seeking behaviors from SQL query logs. Clickstreams [39, 127] supported the analysis of Web traffic and users Web path navigation behavior [51]. Behavior Graphs [39] used state diagrams to visualize search structures on the Web for discovering usage patterns. WebQuilt [216] and Webviz [163] used trees and node-link diagrams to visualize click-streams. Lee *et al.* [127] used parallel coordinates and star fields to visualize user paths and product performance. Wei *et al.* [219] introduced an interactive clustering method to reveal user behavior patterns from Web click-streams data. TrailExplorer2 [187] used stacked bars and pie charts to discover valuable information from large-scale Web click-streams. Lam *et al.* [101] provided users with multiple coordinated views for analyzing Web search behaviors. Shi *et al.* [188] focused on analyzing the dynamics of user loyalty and defection using Web log entries. Stuff I've Seen [74] was developed to provide a unified access to provenance of what a user has seen on their desktop.

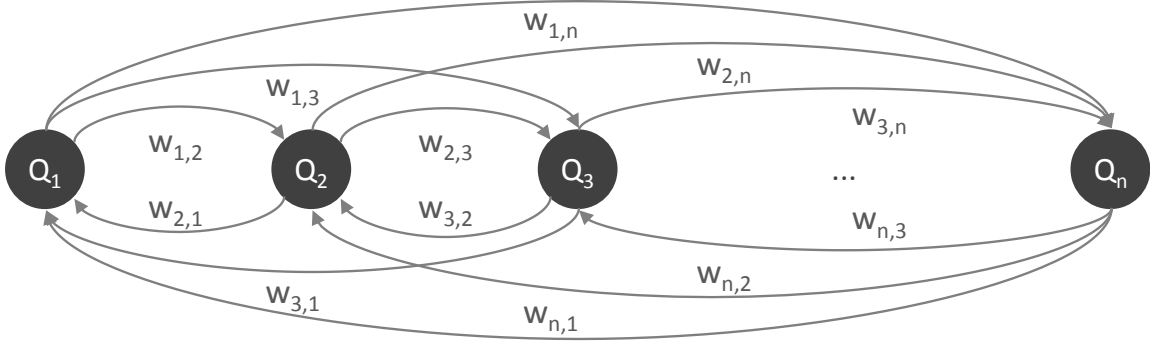


Figure 6.1: In a traditional query-flow graph, each node represents a query and the weight $w_{i,j}$ of the directed edge from Q_i to Q_j represents the likelihood that the two queries are part of the same search session.

6.2.4 Ontology-assisted Visualization

There is a substantial amount of literature on visualizing ontologies and navigating their structures. Many of them can be found in the survey paper [117]. The need to build an ontology of visualization to capture the concepts and characteristics of visualization was discussed in [73]. VisION [170] was developed for categorizing and storing information about software visualization systems. Gilson *et al.* [90] proposed to use multiple ontologies to enable automated generation of visualization. Carpendale *et al.* [40] discussed several research directions in ontology-assisted visualization. Ontology-assisted visualization has also been developed in applications such as biology [8, 64, 99], Big Data [197], social networks [202], geography [132] and so on.

In information retrieval domain, majority of provenance visualization techniques are used and focused upon improving Web marketing, advertising, and merchandising. Therefore, it is essential to develop search provenance visualization techniques to assist users in search. It is also essential to support provenance with ontology-based knowledge support.

6.3 Search Provenance Graph (SPG)

Provenance information is usually stored in a graph-based data structure. For example, five different types of query graphs were defined by Baeza-Yates [12] in the context of web search: (a) *Word Graph* queries contain the same word(s), (b) *Session Graph* queries belong to a same session, (c) *Cover Graph* user clicked on the same

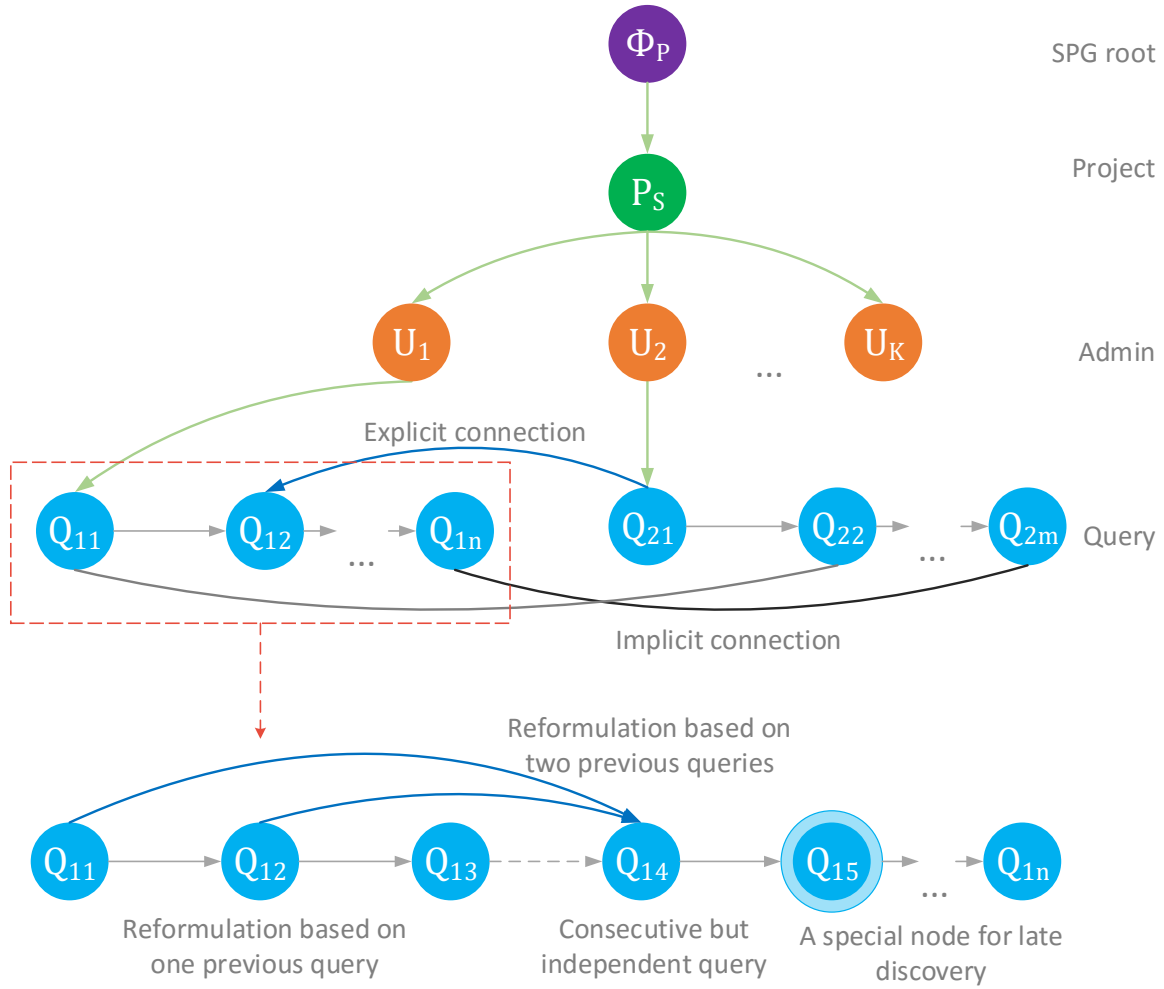


Figure 6.2: In the overall SPG, there are different types of nodes and edges. The nodes are hierarchically organized.

results, (d) *Link Graph* there is a link between the two clicked urls, and (e) *Term Graph* there are k common terms in the contents of the urls.

Boldi *et al.* [25] proposed the *query flow graph* to store statistical and relational information about users' querying behaviors, e.g., interests, preferences, and feedbacks to search results. It is constructed by mining query logs, and it is used for query log analysis, user profiling and personalization, and query recommendation. Figure 6.1 shows a query flow graph, where each node represents a query and each directed edge from query Q_i to query Q_j indicates the likelihood that the two queries are part of the same search session. The likelihood is given by the weight $w_{i,j}$ associated with each edge.

The above mentioned graph-based concepts cannot be directly used for representing provenance of search activities in the context of this work. Firstly, the

provenance in our application is project-based, as described in Section 3.2. It is important for each administrator or a small team of administrators to be aware of the provenance of search activities related to a specific project, while the search activities within a project do not benefit much from the aggregated statistics about many other projects. Secondly, the provenance data need to provide details such as who initiated what queries and when, who is reformulating whose previous queries, etc. We thereby introduced the notion of *Search Provenance Graph (SPG)*.

In a SPG, each query is represented by a node that is associated with attributes such as search query, time-stamp, search results, and selected results. Each edge indicates a connection between two queries. There are two types of connections. An *explicit connection* is typically activated within a search session when a user reformulates a query based on his/her own previous query(s). It can also be activated in a subsequent search session when a user selects another user's search query using the provenance interface as the basis for query reformulation. Hence an explicit connection is inserted by the user interface for search and provenance visualization. An *implicit connection* is inserted into the SPG by a housekeeping program that periodically computes the similarity between nodes in an SPG.

Explicit connections indicate that the administrators formulated queries based on each other, i.e., collaborated well with each other. On the other hand, the degree of implicit connections indicate repeated searches, thus suggesting that collaboration can be improved.

The overall SPG is maintained by the enterprise search system, and it is partitioned into disjoint subgraphs, each of which is associated with a project. Within a project subgraph, there are a list of users (i.e., file administrators), and their search activities are individually recorded.

As illustrated in Figure 6.2, we can thus define the notion of the SPG as follows. The overall SPG is graph $\Phi_P = \{\mathbb{P}, \mathbb{U}, \mathbb{Q}, \mathbb{E}_{pu}, \mathbb{E}_{uq}, \mathbb{E}_e, \mathbb{W}_i\}$. $\mathbb{P}$ is a set of nodes representing different projects. $\mathbb{U}$ is a set of nodes representing different users within each project. If the same administrator is involved in multiple projects, he/she has a distinct node for each project. $\mathbb{Q}$ is a set of nodes representing the queries performed by users. $\mathbb{E}_{pu}$ is a set of directed edges connecting from each project to its users. $\mathbb{E}_{uq}$ is a set of directed edges connecting from each user to its first query in the project. $\mathbb{E}_e$ is a set of directed edges representing explicit connections. $\mathbb{W}_i$ is a set of undirected weighted edges representing implicit connections. The weight of an edge defines the similarity between two nodes. It is in the range of $[\omega_{min}, 1]$ where $0 < \omega_{min} < 1$ is the minimal similarity value for an implicit con-

nection. Any pair of queries with a similarity value below ω_{min} do not have an edge between them. In this project, we set $\omega_{min} = 0.4$, which can be reconfigured.

The connections from the root to projects are implicitly defined since all projects are connected to the root. There are two types of explicit connections in set E_e . A solid directed edge connecting two nodes in Q implies that the destination query is a reformulation of the source query. A node with multiple incoming solid edges results from a reformulation based on more than one previous query. A dotted directed edge indicates that the succeeding query is not based on a preceding node though they are temporally consecutive. An undirected translucent edge is an implicit connection. In computer-generated visualization, the similarity is mapped to the opacity of the edge.

There is also a special type of nodes in Q . It is illustrated in dark shade in Figure 6.2, and represents a set of late discoveries, which are files found by the *Crawler*, rather than the *Query Processing* module.

The overall SPG is dynamically updated during the ongoing search sessions as well as by the background utilities such as the *Crawler* and the housekeeping program for evaluating the similarity between nodes. In order to support such dynamics, many parts of the enterprise search engine have a role to play, which are described in the following section.

6.4 Computing Query Similarity for SPG

Query similarity defines the implicit connections in the SPG. In IR, session and textual features (e.g., TF*IDF, cosine similarity, Jaccard's index, character-level 3-grams and Levenshtein distance) have been used to compute query similarity (e.g., [26,27,91,232]). Fast Algorithms for mining association rules as query similarity is reported in [84]. Among these, the TF-IDF and cosine similarity measure has been widely applied to content matching scenarios. These simple similarity measures can deliver reasonable accuracy. However, it is well understood that many do not include any semantic comparison. Problems arise when synonymies such as "building" and "construction" would be considered as not similar by these measures.

In our system, a search query consists of four fields: *keyword*, *type*, *size*, and *date*. The keyword field must contain at least one search term, while other fields can be set or left empty. In this work, we implemented a similarity function that

computes the score based on all the four fields of the query. In particular, the keyword similarity is computed with the help of the knowledge base of the enterprise search engine.

6.4.1 Keyword Similarity

For measuring the similarity between two strings, we introduced a semantic similarity measurement supported by the *Term Ontology*, which is part of the multi-ontologies described in Section 4.4.1. We propose a similarity algorithm adapted from [167,200]. This algorithm follows two steps:

Step-1: Keyword Graph Construction. Given two queries Q_x and Q_y , we derive two sets $\mathbb{K}_x$ and $\mathbb{K}_y$, which contain the keywords or terms from the queries such that $\mathbb{K}_x \subset Q_x$ and $\mathbb{K}_y \subset Q_y$. Elements of $\mathbb{K}_x$ and $\mathbb{K}_y$ are considered as two disjoint sets and each element can be represented as a node to build a semantic connectivity between the queries. Each node in $\mathbb{K}_x$ is connected to every node of $\mathbb{K}_y$ in the Term Ontology, if there are no connection the distance is considered as ∞ . By omitting the intermediate related nodes and considering only the path length between the nodes representing the original query terms, the semantic network can be considered to be a bipartite graph. The nodes of the first query and second query are the two vertex sets of the bipartite graph where the edge denotes the connection between the nodes as obtained from the semantic network.

The keyword graph is a complete weighted bipartite graph defined as $\Phi_K = \{\mathbb{K}_x, \mathbb{K}_y, \mathbb{W}\}$, where $\mathbb{W} \subseteq \mathbb{K}_x \times \mathbb{K}_y \times [0, 1]$ is a set of weighted edges. This semantic graph is used to compute the similarity between the queries.

For every pair of nodes, $(s_i \in \mathbb{K}_x \text{ and } d_j \in \mathbb{K}_y)$, a function $\lambda : w \in [0, 1]$ computes the weight between s_i and d_j . For each node $s_i \in \mathbb{K}_x$ and $d_j \in \mathbb{K}_y$ we calculate the *shortest_path*(s_i, d_j) a set of weighted edges representing the shortest path between s_i and d_j in the *Term Ontology*. Let $w_1, w_2, \dots, w_k$ be the weights on the k edges connecting s_i and d_j . The connectivity is determined by an aggregate weight between a source node s_i and a destination node d_j . The aggregated weight $\lambda(s_i, d_j)$ is computed as follows:

$$\lambda(s_i, d_j) = \prod_{l=1}^k w_l \rho(l)$$

where, $\rho(l)$ is a step function that moderates the impact of the weight according to connectivity between node n_l and edge l . A commonly-used step function is

$\rho(l) = l^\alpha, \alpha \leq 0$. As $l \in [1, k]$ is an integer variable, the function appears to be stepping-down or horizontal (when $\alpha = 0$). We set $\alpha = 0$, i.e., $\rho(l) \equiv 1$, in this work.

Alternative step functions may also be used. For example, to compute $\lambda(s_i, d_j)$ using only the notion of synset in WordNet, one would have $\rho(l) = 1$ if $l = 1$ and $\rho(l) = 0$ otherwise. To consider all descendants which are k links away from s_i , one would have $\rho(l) = 1$ if $l \leq k$ and $\rho(l) = 0$ otherwise. To consider all descendants in the ‘is-a’ hierarchy, one would have $\rho(l) = 1$ if edges, $1, 2, \dots, l$ are all “is-a” connections, and $\rho(l) = 0$ otherwise. The recursion stops when $\lambda(s_i, d_j)$ becomes insignificant, i.e., less than a threshold value. In this work, the threshold is set as 0.40.

Step-2: Bipartite Matching. Once the complete bipartite graph for the keywords is derived, we are able to apply standard algorithms for optimal matching of the bipartite graph. A matching $W_{opt} \subseteq W$ is a collection of weighted edges such that every vertex of $\mathbb{K}_x \cup \mathbb{K}_y$ is incident to at most one edge of W_{opt} . A matching is perfect if its cardinality is equal, i.e., $|\mathbb{K}_x| = |\mathbb{K}_y|$. Here, the optimal bipartite matching algorithm operates under the simple notion that the connections with higher weights i.e., closely connected keyword nodes, contribute more to the similarity value.

Given the bipartite representation $\Phi_{K_{opt}} \subseteq \Phi_K$, the optimal matching between two keyword or vertex sets can be computed using a maximum allocation algorithm. An optimal maximum allocation algorithm can yield an improved performance, however, as the number of keywords in a search query is a small number, a brute-force algorithm performs reasonably well. The optimal bipartite graph, $\Phi_{K_{opt}} = \{\mathbb{K}_x, \mathbb{K}_y, W_{opt}\}$ where, $W_{opt} \subseteq W$ such that $\Theta(\mathbb{K}_x, \mathbb{K}_y)$ is maximum, is computed as:

$$\forall w_i \in W_{opt}, \quad \Theta(\mathbb{K}_x, \mathbb{K}_y) = 1 - \prod_{i=1}^{|W_{opt}|} (1 - w_i)$$

6.4.2 Similarity: Type, Size, and Date Fields

As mentioned earlier the value of the type, size, and date fields can be set or left empty. Therefore, we introduce two different metrics to compute the similarity of these fields, we call *Behavioral Similarity* and *Value Similarity*. Given two queries, the *Behavioral Similarity* function Ψ measures the similarity based on whether the

fields are populated (set) or are empty (unset) and the *Value Similarity* function Ω measures the degree of match. They are defined as:

$$\Psi(a, b) = \begin{cases} 1 & \text{if } a \text{ \& } b \text{ both are set} \\ 0.5 & \text{if both are unset} \\ 0 & \text{if one is set and another is unset} \end{cases}$$

$$\Omega(a, b) = \begin{cases} 1 & \text{if } a \text{ \& } b \text{ are the same} \\ 0.5 & \text{if there is an overlap between } a \text{ \& } b \\ 0 & \text{if there is no overlap} \end{cases}$$

6.4.3 Aggregate Score

Finally, a function Λ computes the similarity score between two queries Q_x and Q_y by combining the similarity scores of keywords ($\mathbb{K}_x$ and $\mathbb{K}_y$), types (t_x and t_y), sizes (s_x and s_y), and dates (d_x and d_y) fields.

$$\Lambda(Q_x, Q_y) = (1 - \alpha - \beta - \gamma) \Theta(\mathbb{K}_x, \mathbb{K}_y) + \alpha \Psi(t_x, t_y) \Omega(t_x, t_y) + \beta \Psi(s_x, s_y) \Omega(s_x, s_y) + \gamma \Psi(d_x, d_y) \Omega(d_x, d_y)$$

where $0 \leq (\alpha + \beta + \gamma) < 1$. α , β , and γ are scalar constants that defines the influence of type, size and date fields respectively in the similarity measurement. In this project, we set $\alpha = \beta = \gamma = 0.1$, which can be reconfigured.

6.5 Visualization of SPG

Although it is possible to use the basic visual representation of search provenance graph as shown in Figure 6.2, users have to use pop-up windows to retrieve information related previous queries. Frequent interactions can be expensive and, particularly, it is easy to forget what was just shown in the pop-up windows activated. We hence introduced glyphs to enrich each node in a SPG to facilitate a high-level IR at a glance as well as externalize the user's memory about each query.

We began our visual design activities with a series of brainstorming meetings. At first we identified the most commonly-used search query attributes (e.g., keyword/name, type, size, date and search result attributes) to be visualized. We then identified summary features of search results that can help a user refine the

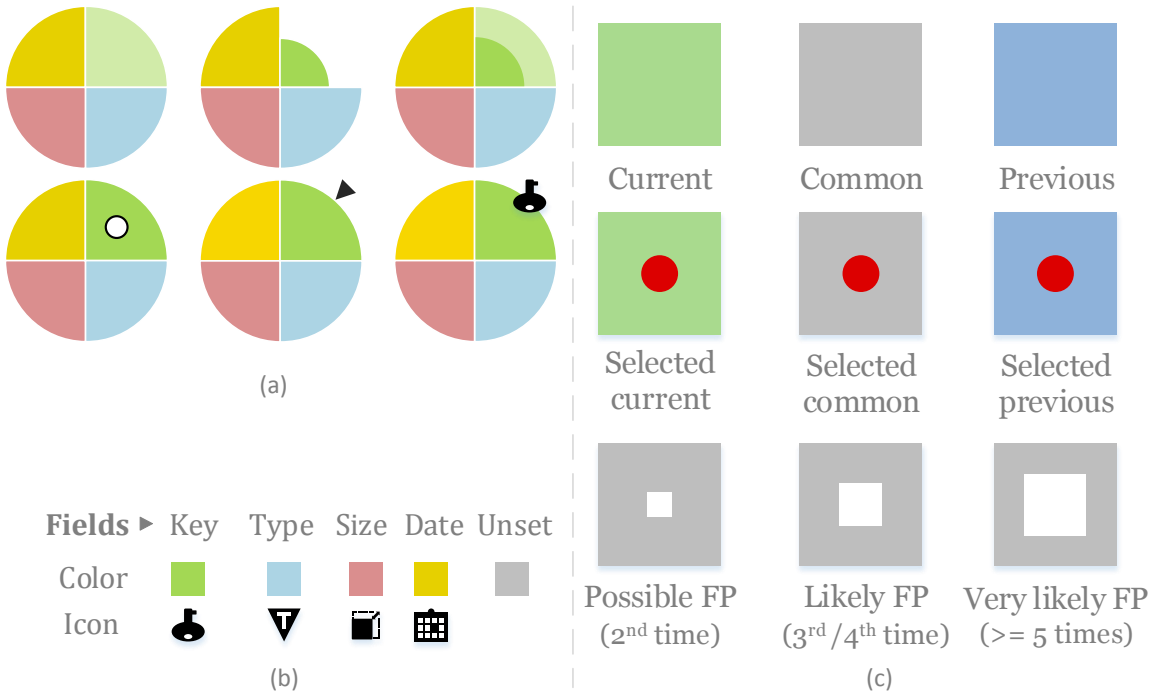


Figure 6.3: Glyph design for search provenance graphs. (a) Six different design options for characterizing a query, (b) the colours and icons used to highlight different query fields, and (c) the glyph components to depict a file in search results (FP: False Positive).

queries. These include the number of newly-found files, how many files were in the previous search, how many files in the previous search are no longer shown, and how many files have already been selected by the user. After several iterations and consultation with our industrial partner a suitable glyph design was chosen based on consideration of occlusion, clarity, visual separation, metaphoric abstraction for aiding learning and remembering, and encoding costs in terms of scalability.

We designed two types of glyphs for representing search queries and search results respectively.

6.5.1 Glyphs for Search Queries

Six different design options for characterizing a query are presented in Figure 6.3 (a). A search query is composed of multiple fields and each field is related to a file system attribute, e.g., *name* or *keyword*, *type*, *size*, and *date*. During the query reformulation a user can set, unset, or change the values of any field. Hence the glyph design should ideally capture the temporal changes of the reformulated queries.

We began our design activities with a series of brainstorming meetings. We produced a table consisting of several optional visual channels for each attribute. After several iterations, a suitable option for each query field was identified on the basis of clarity, pop-out effect, visual separation, metaphoric abstraction for aiding learning and remembering, and encoding costs in terms of scalability. We decided to a k -wedge circle glyph to represent k search criteria. We used logological colour [2] encoding (e.g., kiwi colour for keyword, scarlet colour for size, turquoise for type, and daisy for date). Such logological colour scheme can act as a memory aid. We considered five design options to indicate if a criterion has been added or changed in relation to a previous node as shown in Figure 6.3 (a). After consulting the industrial partner, we decided to use small pictograms to indicate such changes as they can also serve as a reminder about the name of the criterion concerned. These pictograms are shown in Figure 6.3 (b).

6.5.2 Glyphs for Search Results

A set of search results contains much complex information, and designing a summary glyph has been a challenging undertaking. We considered glyphs in the form of miniature treemaps, and found they cannot show much information. We thus decided to focus on some of the basic information needed that may aid the assessment of the effectiveness of a query, and stimulate hypotheses about better query reformulation. Such information includes (i) how many files are returned by the *current* query concerned, (ii) how many are *common* with the previous query, (iii) how many files returned by the *previous* query are no longer in this query, (iv) how many selected files (i.e., true positives) are among each of the above categories of files, and (v) how many files have been returned many times but were never selected (likely to be false positives). Once a file has met a search criteria but is not relevant to the user then this file is classified as *false positives*. When the same file starts to appear in consecutive searches the false positive value of this file is incremented, see Figure 6.3 (c).

The basic glyph design for search results is a macro-glyph consisting of 25 cells, each is encoded with the above 5 pieces of information as shown in Figure 6.3 (c). Following an initial testing, we notice that many initial queries typically result in a large list of files. The 25 cells are not enough. We thus added another type of glyph for showing the overall statistics when there are more than 25 files in the three categories (current, common and previous).

6.5.3 An Example Use of the Provenance Visualization

This example visualizes the search provenance information of four administrators named Charlotte, Julie, Andy, and Jane who collaborated for four working days to create a portfolio of files related to a project for a customer.

The proposed visualizations implement the visual information-seeking principle proposed by Schneiderman in [191]. We first show the overview of the search queries and their similarities (captured in a SPG) using (a) a graph-based view, see Figure 6.4 and (b) a collaboration summary view, see Figure 6.6. Search queries can be selected from either of the views using brushing. For example, the nine queries selected in Figure 6.4 are shown in the glyph-based detailed view, Figure 6.5. The detail information can also be seen in a list view (see the dialogues in Figure 6.5), and using tooltip windows.

6.5.3.1 SPG: Graph View

The graph view shows a subgraph of an SPG which is associated with a project. How one administrator's (focus) search relate to other administrators search who are working collaboratively on a same project. Figure 6.4 shows an example graph view where the four users mentioned earlier searched collaboratively for four working days.

Each search query is mapped to a node of the graph and shown as a dot with different colour coding for different users. Each dot is positioned primarily according to the time when the search event took place. In the central line, we show focus administrator's nodes, here Julie's search queries. The other administrators' nodes, here Charlotte, Andy, and Jane's search queries are shown above as well as below the central line. A link from a central or focus node to a corresponding node above or below the central node is drawn such that if the event has occurred earlier the link will connect to the node placed above else otherwise. This technique significantly reduce the occlusion and cluttering.

In this example, the first two days of searches were performed by Charlotte. It can be easily seen that the user is an inexperienced administrator and has performed many searches some of them consecutively in a short time span. In the following week, the search process was resumed by Julie and Andy who are experienced administrators. In the final day of search, Julie made few more searches, there after Jane joined and performed few queries and final quality checks.

We have used a linear time scale, however, the entire search session is divided into different slots to minimize the empty space when there are no search activity. If there are no search activities for more than an hour (this value can be configured) a glyph is placed to show the elapsed time as a session gap. If the elapsed time is in the range of hours we place a clock icon displaying the amount of hours, and if it is in days then a calendar icon showing the number of days elapsed. A time scale at the bottom shows the dates (below) and time of day (above).

In order to scale the nodes in a given boundary one simple solution is to reduce the size of the dots based purely on the zoom-factor. This solution scales well as the dots can be seen even if they are very small. However, when the dots become too small to be understandable the placement algorithm determines the appropriate size and position of each dot in the time scale and shifts them vertically in order to maintain clarity while preventing serious cluttering and occlusion. The main drawback of this graph view is that it does not show how the administrators searches relate to each other. Therefore, we introduced another visualization technique called *Collaboration Summary View*.

6.5.3.2 SPG: Glyph-based View

Figure 6.5 shows the provenance visualization where glyphs for search queries and search results are shown above and beneath the central line respectively. This example shows the last nine queries performed by Julie, selected from the graph view shown in Figure 6.4. The user formulated the search query by selecting a query performed by Andy on the previous day. The search query is to find a group of building material files with a keyword “building”.

The search query was automatically expanded by the *Query Processor* using the term ontology to include related keywords such as “construction”, “manufacture”, “fabricate” and so on. The search result returned a ranked list of many files and the user selected a few files from the top of the list. The first macro glyph for search results showed that all search results were new (100%) that is the query as well as the results are completely different from the last search the user performed on the previous day. The user selected few files from the search result. Note that the user can see the total number of files in the *Treemap-based focus+context* or *Tabular view* as well as in the tooltip when the mouse is moved over the glyph, reported in Chapter 5.

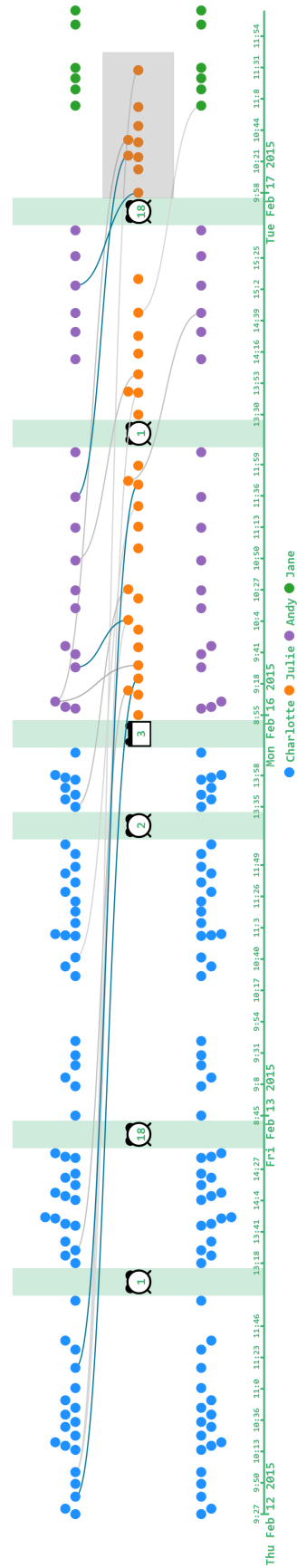


Figure 6.4: This example SPG graph view of Julie’s search provenance window. It illustrates how Julie (focus) collaborated with other administrators. For example, Julie’s fourth query is formulated based on Charlotte and in Julie’s second day of search she formulated two queries based on Andy. The last nine queries from Julie’s search are selected for glyph-based view as shown in Figure 6.5.

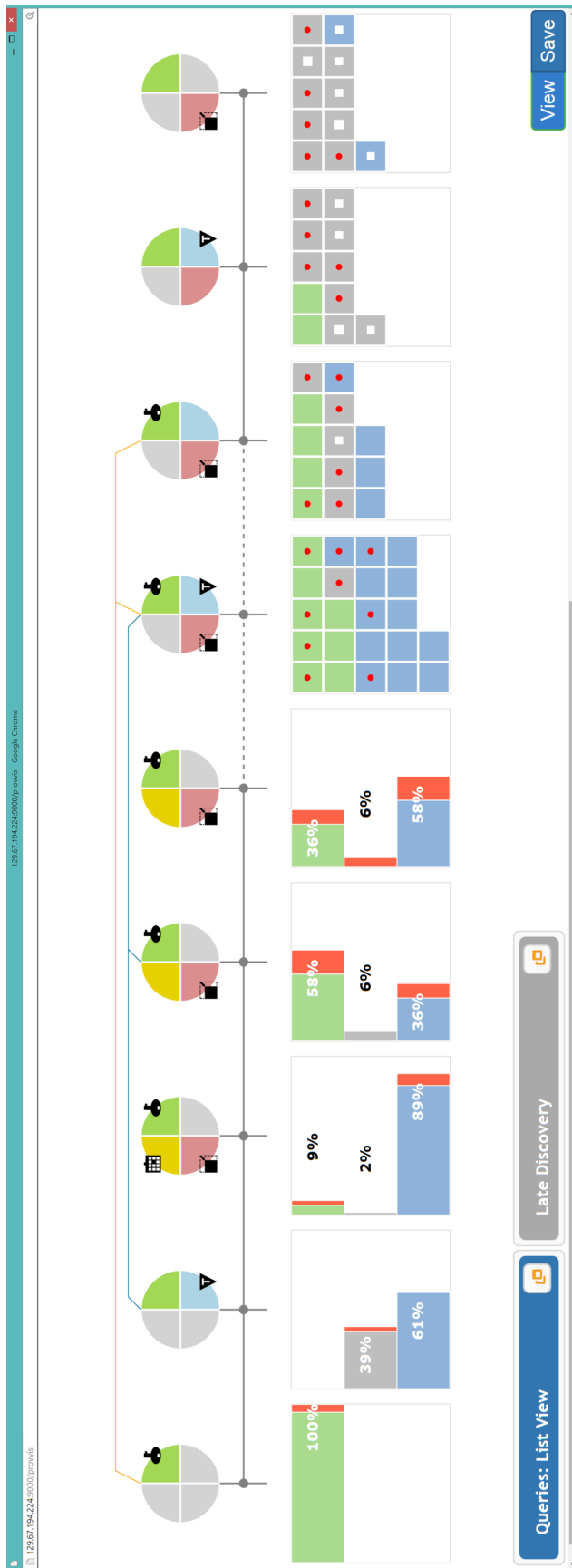


Figure 6.5: An example glyph-based visualization of the queries selected in Figure 6.4. The search queries are shown above the central line, and the search results are shown beneath the central line. Each search query may feature four criteria, clockwise, keyword, type, size and date. An icon appears when this criterion is new or has been changed. The search results are shown in two forms: summary statistics are shown when research results consist of more than 25 files and macro-glyph otherwise. For search results, green represents newly found files, gray represent files common to the current and previous queries, blue represents files appeared only in the previous query, and the red represents the selected files. The background colour of the 'Late Discovery' dialog is gray as there is currently no newly discovered file.

The user then added a type constraint to the search criteria to narrow down the search, and submitted the new query. The second query showed that no new files were returned, and the refined results consisted of 39% common files and 61% files from the previous search disappeared from the list. There were still a large number of files in the search result. The user again made a few selections of files from the top of the list.

The user decided to add one more keyword, “material”, in the query, and removed the type filter. The user also added a size filter particularly to exclude the files smaller than 500KB, and a date filter to exclude the files which are more than three years old. The new results consisted of 9% new files, 2% common and 89% files disappeared. The user continued with two further queries with reformulated search criteria. The reformulated query narrowed down the retrieved result significantly.

The user then decided to formulate a query based on the 2nd and 4th query. The user selected these two queries from the visualization window and specified a logical AND operation to combine their search criteria. The *Search Interface* reformulated a new query and populated the search form. The user was also given an option to review the query or modify it before submitting the new query. The 6th query found 8 new files while 13 files from the previous query now disappeared. The macro glyph changed to a cell-based representation as there were less than 25 files to show.

The user again reformulated the query based on the 6th query that resulted in few false positives and the first query that resulted in a large number of files, some of which were likely to be true positives but were not selected. This combination of search criteria resulted in four new files. The user continued to reformulate the query further. In the 9th query, there was no new file and six files were shown as potential false positives.

All these actions are stored in the *SPG Database*. At any time the user can save a portfolio of the selected files as well as resume the search process from any point.

6.5.3.3 SPG: Collaboration Summary View

The Collaboration Summary view of SPG uses a radial-based layout to visualize the relationships among the administrators searches working collaboratively on a project. The example summary graph shows the subgraph of an SPG which is

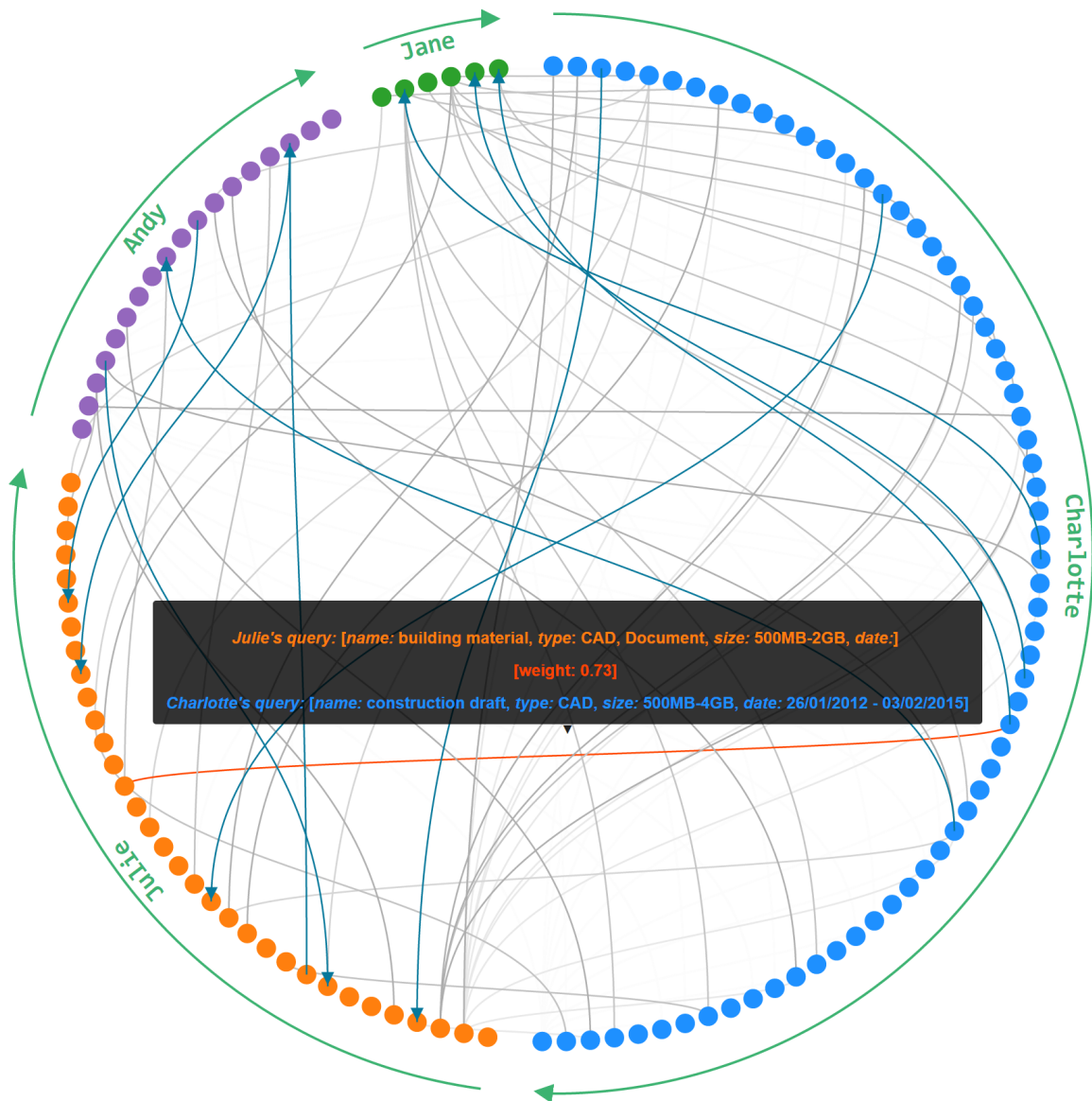


Figure 6.6: This example collaboration summary view of SPG illustrates the similarity between the search queries performed by the administrators associated with a project.

associated with the project. The Collaboration Summary view allows the visualization of their queries relate to each other, and their explicit and implicit connections. The explicit connections are shown in pale-blue colour, and the implicit or system computed similarities are mapped to translucent gray colour. Figure 6.6 presents an example summary graph. Here, for example, Jane performed some last few searches as a quality check and her queries are mostly close to Charlotte's. It can also be seen that Andy and Julie did many collaborative and close searches. On selecting a link, a pop-up window shows the connected queries and the similarity

weight.

In order to scale the nodes in a given boundary, similar to the graph view (Figure 6.4), the size of the dots can be reduced based on the zoom-factor. When the dots become too small to be understandable the radius of the radial graph can be increased and scrollbars can be added if needed.

6.6 Evaluation

6.6.1 Requirement Analysis

At the beginning phase of the project, there were meetings with members of the industrial partner, where the existing document management system and the requirements of the document search system were discussed. Thereafter, the author of this thesis had a week long placement in the enterprise. During that placement, there were a series of meetings with the domain experts, and had the opportunities to familiarize with the existing document management systems, the distributed file system and several file repositories used by different groups across the organization. We discussed about the requirements, the use cases, and the technical challenges of the system. Based on the knowledge gained, the special purpose enterprise search engine to search for files stored across a large distributed system in the organization was developed.

6.6.2 Organization

After implementing the the glyph-based view, the author of this thesis visited the industrial partner to evaluate this visualization facility, together with the latest version of the search engine. During the two-day visit, four domain experts and managers were consulted. They all have an extensive experience of construction management as well as in dealing with large number of documents. The evaluation criteria included (a) the SPG and the glyph design options, and the discriminating capacities of different visual channels, (b) metaphoric mappings for aiding learning and remembering, (c) encoding costs in terms of pixel resolution, (d) importance and suitability of provenance visualization for identifying true positives, false positives, and false negatives, and (e) the usability of the system from the enterprise's perspective.

The author of this thesis had another one-day visit to industry to evaluate if provenance visualizations, e.g., the SPG graph view and the collaboration summary view can assist administrators to perform search tasks collaboratively and efficiently.

6.6.3 Procedure

The author consulted with four domain experts and managers individually as per their convenience. The evaluations were conducted through demonstration of the system and through discussions about the above evaluation criteria. The meeting with each of the experts began with a 15-minute introductory presentation followed by some 20 minutes of open discussion. The discussions were recorded and photographs were taken with the permission of the domain experts. Although the domain experts were aware of the project and the purpose of the meeting, they were still informed of the overall goals of the system, application scenarios, and objectives of the meeting.

6.6.4 Discussion

During discussions, domain experts provided various feedbacks and comments related to the above evaluation criteria. The discussions were lively, and the domain experts were highly enthusiastic in offering their opinions and suggestions. In particular, domain experts asked questions about glyph designs, visual encoding, the search interface and use case scenarios. The idea of visualizing search provenance has been appreciated by all of them. The domain experts learned the visual encoding very quickly, and offered informed suggestions about various design aspects. This suggestion was accommodated into a revised design.

- One expert asked if more query fields are introduced how well it will scale. Therefore, a crush test with 12 query fields to demonstrate the discriminating capacity of various visual channels at different resolutions was showed. The author also explained that the pictograms can change to a small arrow head (glyph design option four) when scalability is an issue, and this option was welcomed by them. A tooltip while hovering over the glyphs to show the query in text has been added later as suggested by the domain experts.

- The domain experts expressed that among all the glyph design options the options which use size or opacity to highlight changes in field may confuse the users, therefore, such options were not included.
- Visualizing the statistics for a large number of search results and attributes was liked by the domain experts. However, they wanted to see the percentage as well as the number of files which are new, common, and deleted. The percentage values were showed in the macro glyphs for search results and a tooltip feature to show the exact number on mouse hover was introduced.
- In order to highlight the selected files, domain experts suggested using a larger and salient red dot, we improved this in our revised design.
- One domain expert asked if there is a way to reformulate the query by combining the previous queries. A feature to perform a logical AND operation was implemented to address this.
- The idea of showing false positives was seen as a useful feature. Such a file may be really important but duplicated, or it may be an outdated version. However, domain experts suggested to use an alternative design option to show the false positive, as the initial design option was to use a halo which was not intuitive and salient enough. In the revised version, the white square design was introduced.
- Domain experts emphasized that showing the explicit connections and implicit or system computed connections in SPG are extremely useful. Therefore, we have introduced different colour coding to aid better visual discrimination capability.

The domain experts made a number of positive comments:

- The circular wedge-based design is preferred by all the domain experts over the alternative design options discussed. They found the choice of pictograms and colours to highlight different query fields intuitive as the pictograms relate to the name of the field.
- The idea of helping stimulating hypotheses about false negatives was much appreciated, as in a large organization most know that some documents are expected at certain locations or business units but not yet found.

- One domain expert expressed that training would be required to learn and memorize the search result glyphs. However, this was not considered to be an issue. Another domain expert mentioned that after using the system over a short period, users can easily remember the fields.
- The domain experts instantly grasped the functionality of the SPG and the visual representations. They found that the collaborative graphs are helpful to avoid repetitive work or learn from previous searches. As all the search actions are stored and can be visualized later on, the SPG can assist the lead administrator to perform performance review, assess work quality and train new users, commented by the experts. As sometimes one administrator might have to step into a project to resume a search from any point past, the graph view would allow one to overview the situation quickly.
- The domain experts found that the provenance visualization enables quick interpretation of the search criteria, and improved quality of the retrieved results.
- One domain expert said: “This is a nice and simple representation to show the high-level overview of the reformulated queries and the relationship among the results. Showing the search result size, and the selection within and comparing with the previous search is nice.” Another domain expert commented: “Showing the relationship in terms of new, common and deleted results in the form of a glyph is a great idea, as it can guide the user by giving an idea to effectively reformulate a query.”
- Another domain expert commented: “I find it quite useful, it makes sense, user can easily get an idea about whether they need to refine the results further, for example, if there are few new files in a search result the user may reformulate the query.”
- Another expert commented: “It is a time consuming task to retrieve all the relevant documents related to a search. Many times during a search the user gets lost and performs the same search repeatedly. The use of this new visualization could eliminate this problem, and could therefore prove to be very useful. I can definitely see a business application of this search interface and visualization in an engineering enterprise where documents are so important in the same time very difficult to deal with it”.

This second evaluation also prompted two domain experts to carry out business cost-benefit analysis. We broke the search processes detailed action steps, including capture a file and place it in a server, enter a record in a database, database maintenance, search by a local administrator based in a project, search by central administrators with and without the aid of visualization, packaging selected search results, and so on. The detail description of the business cost estimation can be found in Section 3.4.

The industrial partners realizes the important role and potential benefits of the visualization-assisted search solution to handle their large-scale data generated throughout different project life cycles which has been highlighted in their recently published magazine [125].

6.7 Conclusions

In this chapter, we have presented a visualization-assisted search solution to support enterprise search. We have developed a prototype system featuring both an enterprise search engine and a set of visualization capabilities. The development of the provenance visualization has benefited particularly from the knowledge-base of the search engine. This demonstrates that visualization can and should play an integral role in an enterprise search engine. Our industrial feasibility study has shown that visualization can impact on industrial applications by improving working practices in the real world. The work helped the enterprise to realize the important role of visualization in search operations at an industrial scale. It provided the basis for establishing a cost-effective working practice, and developing new software for supporting such routine but intensive operations. From this work, we may draw a generalized conclusion that visualization can be used to help users identify false positives and reason about false negatives in search processes. Search provenance visualization provides an effective means for users to reformulate search criteria, facilitating external memorization in a search task that may be performed over a short period and by multiple users collaboratively.

7

Conclusion

Contents

7.1 Summary of Contributions	143
7.2 Summary of Future Work	148

This thesis set out to introduce information visualization techniques for supporting an enterprise search engine, particularly to assist administrators to perform search tasks efficiently and collaboratively in a large-scale industrial setting. Searching through a vast and evergrowing information space is a time consuming task particularly dealing with the inevitable inconsistencies in search results being returned by enterprise search engines. While the role of visualization in a large-scale industrial search has not yet been explored, this industrial feasibility study has been formulated in collaboration with our industrial sponsor, Laing O'Rourke.

As part of our study of the visualization capabilities, we developed a special purpose enterprise search engine in Chapter 4. This chapter describes the core components needed to design and integrate the visualization components presented later in Chapters 5 and 6. More importantly, through the design and development of the enterprise search engine and the visualization components, we were able to devise strategies to investigate the research objectives stated in Chapter 1.

7.1 Summary of Contributions

We state the contributions of this thesis through focusing on the objectives from Chapter 1.

Objective 1. To investigate how to overlay search results on top of a search space in the context of large-scale distributed file systems to facilitate focus and context visualization.

Using a traditional representation of search results (e.g., a text view), it leads to a greater difficulty and time to devise hypothesis about whether a search is successful or not (e.g., false positives and false negatives) or how to refine search specifications. It is essential to show an overview of the search results and their distribution in the search space that can give a contextual information.

In Chapter 5, we introduced a novel *Focus and Context* visualization technique to overlay search results (as focus) and their distribution in the search space (context). The trivial representation of the search results as dots overlaid on top of treemaps causes a greater difficulty in identifying these dots. An improved design using a file system glyph can enrich the visual representation as well as show the multivariate information, however, overlaying such glyphs on top of a dense visual context resulted in occlusion of the context.

We proposed to use cyclical property animation of these glyphs to minimize the occlusion and to allow both focus and context to be shown simultaneously. The idea of overlaying glyphs on top of a treemap [111] and to apply cyclic animation to glyphs to minimize occlusion was studied in this work for the first time. There has not been any previous study or guideline on how to minimize such occlusion or how to improve the perceptual difficulties in glyph animation. Therefore, we designed a set of transfer functions with carefully assigned interpolations (e.g., to enable the human visual system perceive the information) and three novel empirical studies in the form of control experiments and a focus group study.

The control experiments investigated the perception of the animated glyphs: (a) the effects of different temporal transfer functions on focus, (b) the effects of different temporal transfer functions on context information, and (c) the effect of animation on the just-notifiable-difference (JND).

The focus group study qualitatively evaluated the approaches, namely (a) static dots, (b) cyclically-animated dots, (c) static glyphs, and (d) cyclically-animated glyphs. We discovered that nearly all the participants preferred animated dots, the second preference was static and animated glyphs, and most of the participants did not prefer static dots. We also observed that the participants could learn the visual encoding of the glyph quickly. The proposed focus and context visualization proved to be useful in quickly interpreting the search results, their location in the file system, and identifying the true positives or filtering the true negatives

by interactively navigating through file system hierarchies.

Objective 2. To develop a search provenance management technique and to introduce provenance data visualization techniques.

In an industrial scale search setting (e.g., at Laing O'Rourke), it is desirable for a team of administrators to perform search tasks efficiently to create a portfolio of relevant files to be delivered to stakeholders (e.g., customers). Where such routine activities are carried out on a regular basis, it is essential for the administrators to collaborate and effectively formulate/reformulate queries by observing and learning from previous mistakes and good practices.

Graph-based provenance models (e.g., Query Flow Graph [25]) capture the users search provenance data (e.g., from Yahoo search), however they are mainly useful for research in the Web search domain. These models cannot be adapted in the context and with the requirements of an industrial scale search. Therefore, in Chapter 6 we invented a novel graph structure called *Search Provenance Graph (SPG)* for systematically managing the search provenance information in an enterprise search engine.

We also adapted an improved similarity computation algorithm to calculate the similarity between the queries in the SPGs, as commonly used similarity measurement algorithms overlook the semantics in such computation (e.g., TF-IDF [112] or cosine similarity [182] or even the most recent work e.g., [26,27,91,232]).

We proposed three visual representations for visualizing the information stored in the SPG: *Graph View*, *Glyph-based View* and *Collaboration Summary View*. The evaluation of the visualization components in collaboration with a management team at Laing O'Rourke highlighted some potential benefits of such visualization. First, the evaluation confirmed that the graph view is effective in displaying the search history and relation between the administrator's search queries. Second, the evaluation confirmed that the collaboration summary view showing explicit and implicit connections can effectively visualize the quality of collaboration in a project. Finally, we learned that the glyph-graph representation of SPG can aid the assessment of the effectiveness of a query and stimulate hypotheses about better query reformulation.

Objective 3. To develop a special purpose enterprise search engine for enabling the integration of visualization facilities which cannot be done by a general purpose or open source enterprise search engine.

It is not straightforward to develop and integrate the proposed set of visualizations, the (a) search result visualization and (b) search provenance visualization, with a general purpose or open source enterprise search engines. Although such search engines provide an efficient index management and storage facilities, they did not implement any mechanism to manage search space and search provenance information much needed for the integration and investigation of the visualization capabilities presented in this thesis.

In order to facilitate the development and integration of the visualization components, a multi-ontology based enterprise search engine was developed in Chapter 4. Some additional details of this search engine can be found in Chapter 3 and Appendix A.

A realtime scanning and search on a large-scale distributed file system would not be feasible given the scale and nature of a distributed system. For this reason, this special purpose search engine implemented a mechanism to crawl and index the distributed file system. This indexed information provided the file system information and structure needed by the treemap to visualize the search space. The actual search queries are also processed with the index.

Additionally, the ontologies of the search engine supported the integration of search provenance information in the SPG and the computing of the semantic query similarly in SPG.

Finally, the ontologies provided the basis for query enrichment and the implementation of the active learning mechanism using users relevance judgement information.

Objective 4. To investigate how machine learning techniques can be deployed in an enterprise search engine to improve the precision and recall of search results.

The users' relevant feedback information has widely been used for machine learning purposes in Web-search domain. However, unlike Web-search, the enterprise search has not yet taken full advantage of such machine learning techniques [41, 123]. Although commercial systems (e.g., Google Enterprise) might incorporate such mechanism, the exact methods used by these systems are unpublished. The role of an active learning technique to improve the retrieval efficiency of an enterprise search engine measured in terms of precision and recall has been investigated for the first time in Chapter 4.

We adapted the Rocchio's [175, 181] algorithm to calculate an aggregated Relevance Feedback Index (RFI) and developed a novel active learning mechanism

to train the ontologies. Moreover, we also implemented ontology-assisted: (a) knowledge-based query enrichment for improving recall, and (b) ranking of search results for improving precision.

Our experimental study demonstrated that the active learning of the ontologies using users' relevant feedback information improve both recall and precision of the search queries that the search engine relied on to achieve an interactive performance in search. The experimental study also proved that training played an important role in improving the precision and recall, and will be essential and effective in practical deployment.

Objective 5. To evaluate all the techniques developed in this project in conjunction with the industrial partner.

There are several ways to evaluate the techniques proposed in this thesis, e.g., to conduct user studies, to do an industrial interview, and to perform a business cost estimation. Objective 1 was evaluated using user studies that are described in Chapter 5. Objective 2 was evaluated using industrial interviews that are described in Chapter 6. In Chapter 3, we discussed the business cost estimation of the prototyped visualization assisted search system which was done in collaboration with our industrial partner, Laing O'Rourke. The estimation projected the potential benefit of the Visualization-assisted enterprise search engine in terms of saved time of the administrators.

In order to thoroughly approve the system, it would be essential to deploy the system in an industrial setting. It has to be recognized that any thorough analysis can only be done after deployment, testing, and usage over a long period of time in an industrial setting. Such analysis would take many years to achieve. For that reason, such study is not feasible given the scope and duration of this thesis work. With the business cost estimation, it has now become clear that the benefit of this visualization technique is to save time, allowing the administrators to devote more valuable cognitive effort to develop hypothesis, and improve decision making.

This project is the first step to introduce visualization in industrial scale enterprise search. This research has moved one step further towards the overall aim which is to provide industry a visualization assisted search technology that can address the problems emerging from handling large number of files in a cost effective way.

The feasibility study helped our industrial partner, Laing O'Rourke, to realize

the cost effective means to manage and search a large-scale information infrastructure along with the important role of visualization in search operations. As organizations are relying more on digital infrastructure, the demand for scalable file management and search will continue to grow. Our work has provided the basis for establishing a cost-effective working practice, and developing new software for supporting such routine but work intensive operations. The aim has long steps to achieve and is waiting for software development, changing working flow, testing, and field trial in an industrial setting.

7.2 Summary of Future Work

The work presented in this thesis opens up a number of interesting research avenues around the industrial scale enterprise search area.

While the construction companies are preparing to implement the new BIM requirements, deploying an enterprise search engine is essential to make all the files stored in the organization searchable. Building on the feasibility study with Laing O'Rourke, we believe that the software prototype is ready for industry to take over and develop an operational software. The design concepts produced in this work can be featured in the operational software and can be deployed in Laing O'Rourke. We estimate that probably it will require five person years to develop a industrial grade operational enterprise search engine for Laing O'Rourke. It is to be noted that open source enterprise search engine, e.g., Lucene took more than 15 years to develop up to Version 5. We are also in discussion with our industrial sponsor about disseminating the source code of the search engine and visualization software.

When the enterprise search engine is deployed in a real environment, it will require the company to introduce new processes and reorganize its existing workflows. In a large enterprise such as Laing O'Rourke, creating an operational team and introducing new workflows is expected to be a complex process. Once the workflows are reorganized the performance testing and improvement of the system can be done.

As for the visualization side of work is concerned, one important future work would be investigating the usage of spatially ordered treemap [224] to display the search space. There needs to be better understanding of the user interface, and perception of the visual design, which is again possible once a operational software

is deployed and the organization is ready to adopt the new workflows. We suggest that future studies may obtain the user performance data through task-driven experiments in controlled settings. As such task performance may be related to several aspects, such as accuracy, speed, learning, memorization, and irritation. Inevitably it would require many empirical studies in order to obtain statistically-significant data.

Another essential element for future work would be content-based retrieval in addition to metadata based retrieval. Although content-based retrieval has been extensively discussed in the literature and implemented by open source enterprise search engines, however, for efficient information retrieval and ranking different search strategies need to be developed for different types of files. For example, Excel or spreadsheets need different search strategies than text or documents, similarly image, video, CAD designs, and so on need different search strategies. At the same time the ontologies should support content retrieval, which would bring new complexity due to the size and computational requirements. Given the number of possible contexts (e.g., name-space, location, personalization, different domain specific knowledge, and so on) that can be taken into account, the potential application of ontologies in enterprise search engines is vast, and such system may need many years to evolve.

Appendix A

Search Index Management

Contents

A.1 Crawler	151
A.2 Managing an Index	152
A.3 Managing a Dynamic Index	160

In this appendix, we describe the *Index Management* module of the enterprise search engine. We discuss the key data structures and the algorithms that are used to create an index of a large number of files that can be searched quickly, updated efficiently, backup in a disk, and restored from the disk. We adapted the techniques describe in [136] and Lucene ¹. Thereafter, we designed a simple index management data structures which easily integrates the search engine modules (e.g., *Query Processing*, *Result Processing*, *Active Learning*, *Crawler*, *SPG Management*, and *Ontology Management*) of the VAESE system. The index management module was implemented in C++.

A.1 Crawler

When we search for the files in a filesystem using keywords, a naive approach would be to use a crawler or GNU/Linux “find” command to sequentially scan each file of the filesystem for the given keyword. This approach has two main drawbacks- (a) it does not scale well in a large-scale distributed file system due to the size and the nature of the distribution and (b) scanning the file system in real-time for each search operation is time consuming. Therefore, using the

¹lucene.apache.org

GNU/Linux “find” command it is impossible to search files in a large-scale filesystem at an interactive rate.

The solution to this problem is to build a cache or index of the files and folders of the distributed file system during the crawl process. Index enables lower response time to frequent queries and reduces the load on the system by eliminating the slow sequential scanning process. In a large enterprise, new files are being created and existing files are updated or removed frequently. Therefore, the freshness of the index is another essential requirement.

Crawler is an agent or group of agents that gathers and scopes the contents of the file system that need to be indexed. The crawler acts as an interface between the distributed filesystem and the index. The protocols to access different storage systems are defined in the crawler software. The crawler agent scans through the entire distributed file system in an automated manner and at a predefined scheduled interval, much like GNU/Linux utility “Cron”. The crawling process is incremental so that it fetches only the changed files since it was run. A database is used to check time stamp only to avoid re-indexing the files that have not changed since the last indexing.

A.2 Managing an Index

Searching is a process of looking up keywords or terms in an index to find the files where they appear. We consider a number of factors when developing the index management module, the ability to: (a) index large quantities of files, (b) search files in a large index quickly, (c) frequently update the index, (d) backup the index in a disk, and (e) restore the index after system failure or scheduled maintenance. The index management module periodically reconstruct a new index from scratch while the old one is still available for querying, also known as static indexing [136]. After the new index is ready the old index is replaced.

A.2.1 Term Processing

Each crawled file is added to the index, however, before adding a file to the index its metadata values are processed. Metadata values are extracted from the files and processed to produce a stream of tokens. The tokens are then normalized and lowercased. It is also desirable to remove all the stop words, which are frequent but meaningless (e.g., a, an, the, in, on, and so on, in English text). The tokens

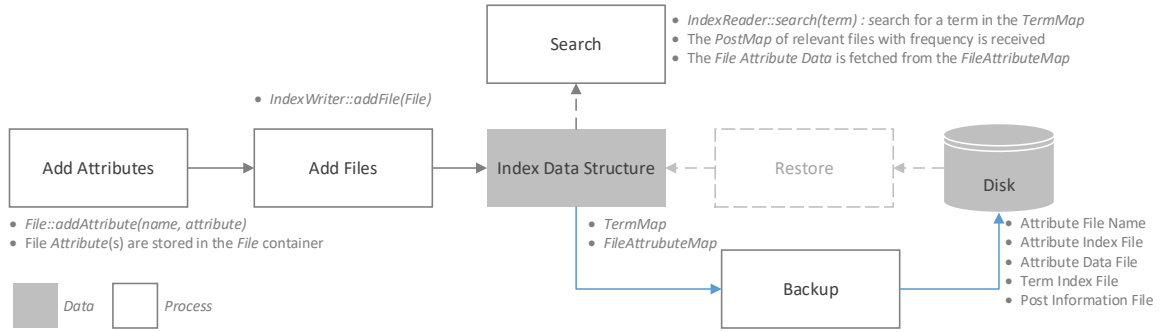


Figure A.1: An index creation and search process. Here, searches happen in the *Inverted Index* (*TermMap*) and the *File Attribute Data* (*FileAttributeMap*) stored in memory.

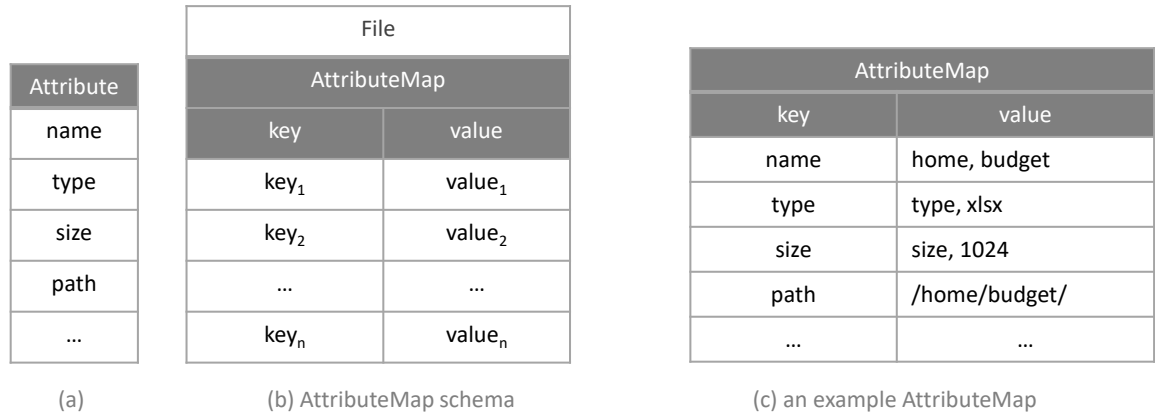


Figure A.2: (a) A file *Attribute* structure holds all file attribute names, e.g., file name, type, size etc. (b) *AttributeMap* holds all the attribute name and value of a file as a key-value pair. (c) In this example, a file `/home/budget/budget.xlsx` of 1024 byte size is indexed in a *AttributeMap* container.

are then reduced to their roots using Porter Stem filter and/or the *Term Ontology*. For example, for a file `/home/budget/budget.xlsx` of 1024 byte the derived tokens are “home”, “budget”, “xlsx”, and “1024”. Detailed description of the term processing steps can be found in [136].

A.2.2 Index Data Structure

The processed tokens along with actual file and its metadata information are then added to the index. Figure A.1 illustrates the steps involved in index (static) creation, search, and backup. The index is formed with two main data structures:

- (i) File *Attribute Data* is a data structure holding the crawled files and their

FileAttributeMap	
fileId	attributes
fileId ₁	...
fileId ₂	...
fileId ₃	...
...	...

(a) FileAttributeMap schema

FileAttributeMap	
fileId	attributes
0	[name: home, budget], [type: xlsx], [size: 1024], [path: /home/budget/], ...
1	[name: home, budget, finance], [type: docx], [size: 2318], [path: /home/budget/], ...

(b) an example FileAttributeMap

Figure A.3: (a) File Attribute Data– each file is assigned with a unique identifier (`fileId`), and its attribute values are stored in a `FileAttributeMap`. (b) In this example, two files `/home/budget/budget.xlsx` and `/home/budget/finance.docx` are assigned with `fileId`: 0 and 1 respectively.

TermMap		
term	fileId	frequency
t_1	fileId ₁₁	tf ₁₁
	fileId ₁₂	tf ₁₂
	...	...
t_2	fileId ₂₁	tf ₂₁
	fileId ₂₂	tf ₂₂
	...	...
...	...	...
t_n	fileId _{n1}	tf _{n1}
	fileId _{n2}	tf _{n2}
	...	...
	fileId _{nk}	tf _{nk}

(a) TermMap schema

TermMap		
term	fileId	frequency
budget	0	2
xlsx	0	1
home	0	1
	1	1
finance	1	1
docx	1	1
...	...	...

(b) an example TermMap

Figure A.4: (a) Inverted index– is stored in the `TermMap`. It contains the term (as key) and the files it is appearing on with its frequency in the post map (`PostMap`). (b) The example inverted index is created from files illustrated in Figure A.3 (b).

attribute (metadata) information, see Figure A.3. An associative container named `FileAttributeMap` holds that information in memory.

(ii) Inverted index is a data structure holding mapping from terms or keywords

to a set of file containing it, see Figure A.4. An associative container, `TermMap`, holds that information in memory.

Each file and its attribute information are processed to create an index which follows three steps:

1. Each file and its attributes are processed to a `File` container data structure. The `File::addAttribute(name, attribute)` adds each file attribute to a `AttributeMap` of the `File` container, see Figure A.2. The function `File::addAttribute(name, attribute)` (A.1) does this job. Each attribute has a name (key) for identifying it, attribute value, and additional information (e.g., the attribute data type, storing policy, and so on). Each attribute can be flagged to indicate if it is to be indexed or not.

For an input file once the `File` container is created then it can be added to the index.

2. The file attribute data stores all the incoming files and its attributes (i.e., the `File` containers) in a `FileAttributeMap`, see Figure A.3. The function `IndexWriter::addFile(File)` (A.2) does this job.

Figure A.3 (b) illustrates an example `FileAttributeMap` that is constructed from two input files `/home/budget/budget.xlsx` and `/home/budget/finance.docx`. Here, the size attribute is not indexed (flagged out), for that reason size attribute (tokens) does not appear in the example inverted index.

3. In the mean time, the function `IndexWriter::addFile(File)` (A.2) updates the file information to the inverted index, i.e. `TermMap`. The inverted index contains the terms and each term entry contains its post information. The post information of a term contains the files the term appearing in and its corresponding frequency. In other words, the post information contains a unique file identifier (`fileId`) as key and frequency as a mapped value. The post information is stored in `PostMap`.

The tokens extracted from input files are used as lookup keys instead of treating them as the central entities. The `TermMap` shown in Figure A.4, is a associative container that store each term as a key and its corresponding `PostMap`. This data structure allows quick lookups for a term and its corresponding files with frequency value.

Figure A.4 (b) illustrates an example of how a `TermMap` constructed from two files stored in a `FileAttributeMap` (Figure A.3 (b)). As the size attribute is not indexed, hence, the tokens of size attributes (“1024” and “2318”) do not appear in the `TermMap`.

Algorithm A.1 `File::addAttribute(name, attribute)`

Description: File attribute names and values are added to a map..

- 1: define `AttributeMap <name, attribute>`
 - 2: `AttributeMap[Attribute.name] ← attribute`
-

Algorithm A.2 `IndexWriter::addFile(File)`

Input: File container that holds attributes of a file in `AttributeMap`.

Description:

- 1: define `PostMap <fileId, frequency>`
 - 2: define `FreqMap <term, frequency>`
 - 3: define `TermMap <term, PostMap>`
 - 4: define `FileAttributeMap <fileId, Attribute>`
 - 5: **for** iterate `AttributeMap` **do** ▷ File container holds the `AttributeMap`
 - 6: `data ← data + AttributeMap.second.data`
 - 7: `FreqMap ← parse(data)`
 - 8: **for** iterate `FreqMap` **do**
 - 9: `term ← FreqMap.first`
 - 10: `frequency ← FreqMap.second`
 - 11: **if** `TermMap.find(term)` **then**
 - 12: `PostMap[fileId] ← frequency`
 - 13: `TermMap[term] ← PostMap`
 - 14: **else**
 - 15: `TermMap[term][fileId] ← frequency`
 - 16: **end if**
 - 17: `FileAttributeMap[fileId] ← AttributeMap.second`
 - 18: **end for**
 - 19: **end for**
 - 20: `fileId ← fileId + 1`
-

A.2.3 Backup and Restore

It is important to provide a mechanism to save the index stored in memory (RAM) to a disk in order to quickly recover the index after an unexpected system failure or

Attribute Name File (.anf)
of attributes
name
type
size
path
...

Figure A.5: The format of Attribute Name File (ANF). The `Attribute` data structure is saved in this file.

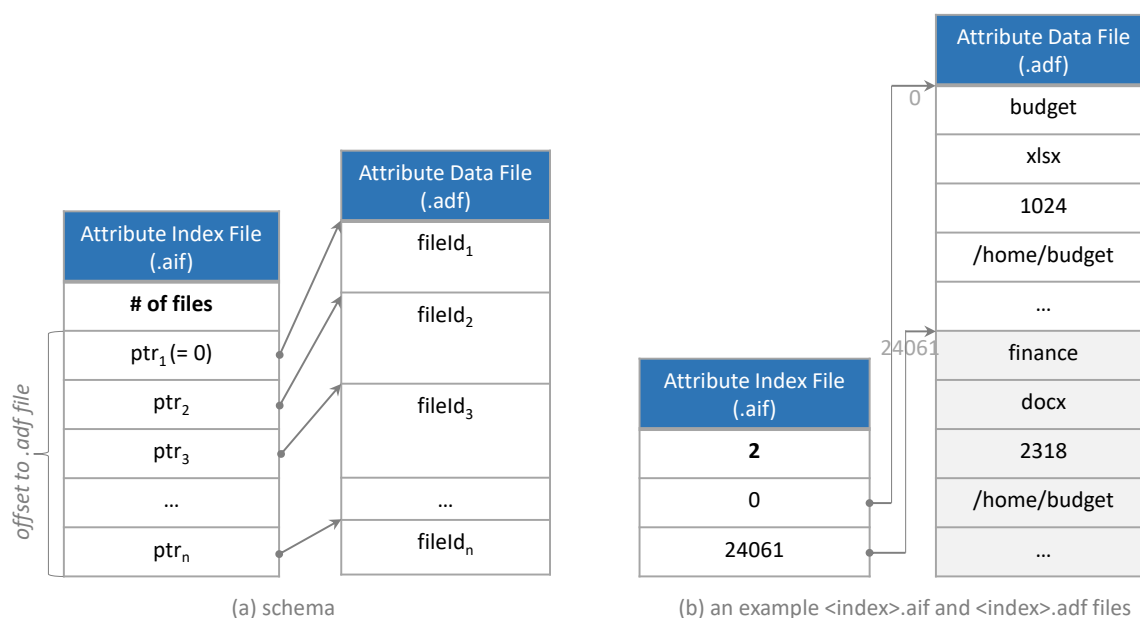


Figure A.6: (a) The formats of Attribute Index File (AIF) and Attribute Data File (ADF). These files store the `FileAttributeMap`. (b) This example shows an AIF and an ADF storing the `FileAttributeMap` illustrated in Figure A.3 (b).

scheduled maintenance. The data structures are stored in the disk in an optimized way so that the data can be restored quickly from the disk. The detailed formats of the files holding the index information are explained in this section.

The file attribute names stored in the `Attribute` structure are saved in an Attribute Name File (ANF, with extension .anf). The ANF contains the number of attributes followed by the name of the attributes (e.g., file name, path, size, etc.) that are being managed in the index. The format of the ANF file is illustrated in

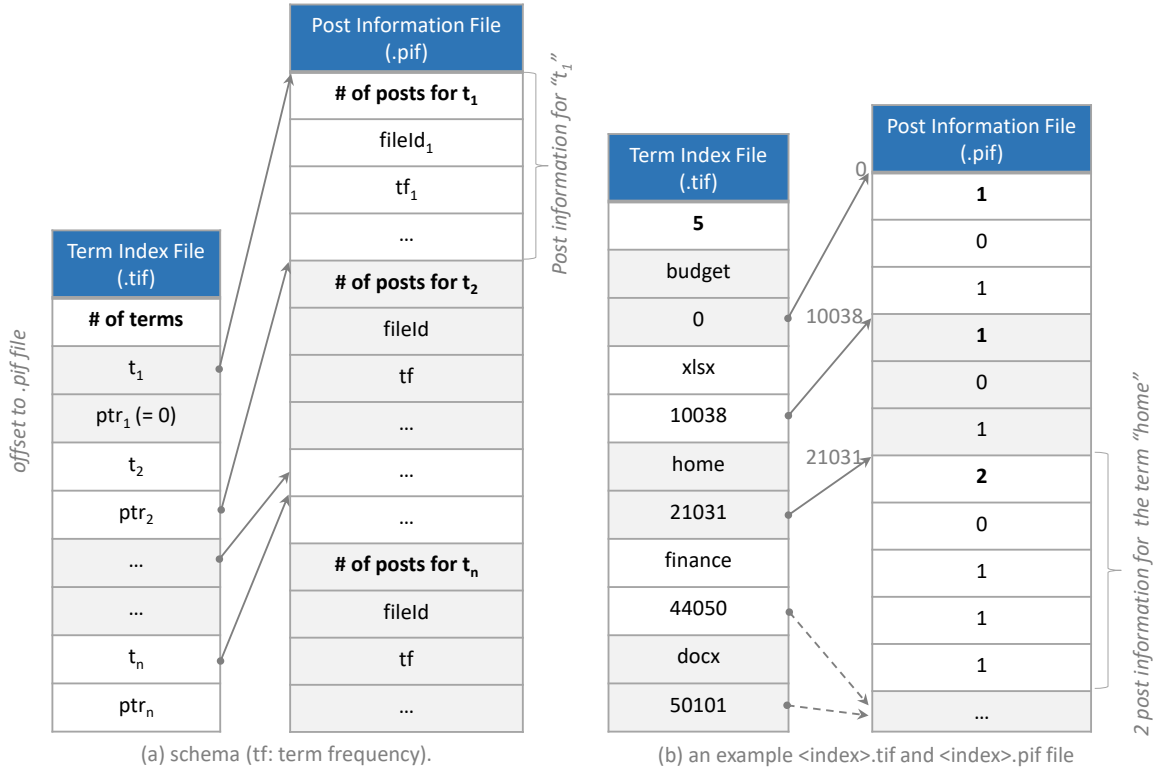


Figure A.7: (a) The formats of Term Index File (TIF) and Post Information File (PIF). The inverted index information, i.e., the `TermMap` is saved in the disk using these two files. (b) This example shows a TIF and a PIF storing the `TermMap` illustrated in Figure A.4 (b).

Figure A.5.

The file attribute data, i.e., the `FileAttributeMap` is saved in two files: Attribute Index File (AIF) and Attribute Data File (ADF). The AIF contains offset information, which is used to resolve file number (`fileId`) to exact position in the ADF file for that file's stored attributes, thereby, supporting quick random access of the files. The ADF contains the actual contents of the file attributes. The formats of AIF and ADF are shown in Figure A.6.

The inverted index, i.e., the `TermMap` is stored in two files: Post Information File (PIF) and Term Index File (TIF). The post information (`PostMap`) of the terms are stored in the PIF. All the terms, tuples of term name and corresponding offset to PIF are stored in the TIF. The formats of PIF and TIF are shown in A.7.

While it is important to save the in-memory index to the disk, it is important to recover the information stored in the files after any unexpected system failure or scheduled maintenance to provide querying facility. The `Attribute`, `FileAttributeMap`, and `TermMap` data structures can be built from the files

holding the information in the disk as a backup. In case of corrupted backup files, the entire distributed file system has to be crawled to rebuild the index from scratch.

A.2.4 Search

During search, the files that matches search keyword are retrieved along with their attribute information. The function `IndexReader::search(term)` (A.3) returns the files that matches a given term. This algorithm first searches the term in the `TermMap`, if the term is present then its corresponding `PostMap`, i.e., the relevant `fileId(s)` with frequency are returned.

Algorithm A.3 `IndexReader::search(term)`

Description:

```

1: define PostMap <fileId, frequency>
2: if TermMap.find(term) then
3:   PostMap  $\leftarrow$  TermMap.second
4:   for iterate PostMap do
5:     fileId  $\leftarrow$  PostMap.first
6:     frequency  $\leftarrow$  PostMap.second
7:     Attribute  $\leftarrow$  FileAttributeMap[fileId]
8:   end for
9: end if

```

A.2.5 Search in a Large Index

The index grows monotonically along with the incoming number of files, therefore it is inefficient to maintain a large index in memory. We can keep deleting the old entries to make free space for the crawler to add new entries. However, in many situations it is not desirable to delete the entries from the index. This problem can potentially be addressed by: (a) maintain the inverted index (`TermMap`) in memory and (b) access the large part of the index from disk, e.g, access the file attribute data from the AIF and ADF. While searching for a term, the relevant files are first searched in the inverted index. Thereafter, the corresponding file attributes are obtained by randomly accessing the files in the disk. Figure A.8 illustrates the steps involved in searching in a large index. Figure A.9 illustrates an example search

APPENDIX A. SEARCH INDEX MANAGEMENT

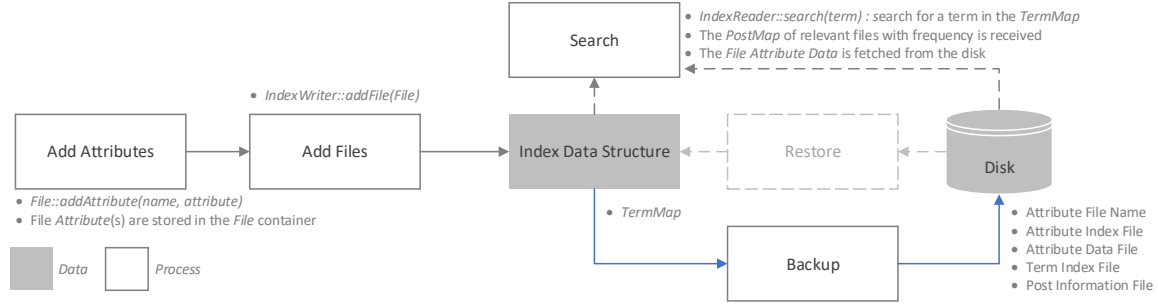


Figure A.8: Index management using in-memory and disk data. Here, a part of inverted index (*TermMap*) is loaded in memory. The keywords are searched in the *TermMap* and the corresponding file attributes are accessed from the PIF, AIF, and ADF stored in the disk.

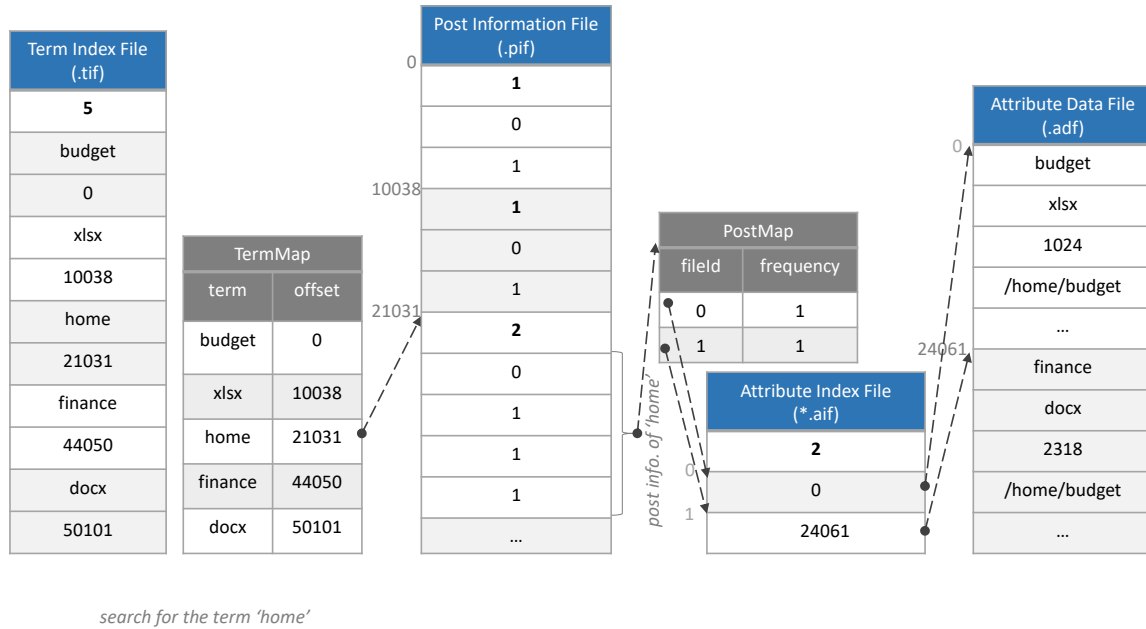


Figure A.9: This example illustrates a search to retrieve all the files related to the keyword “home”. The inverted index (the *TermMap*) is accessed from memory and the file attribute data is accessed from the files stored in the disk.

scenario. The experimental data demonstrates the scalability and effectiveness of this proposed technique, see Section 4.5.3.

A.3 Managing a Dynamic Index

In static index management facility (described in Section A.2), a new index is periodically reconstructed from the scratch while the old index is still available for

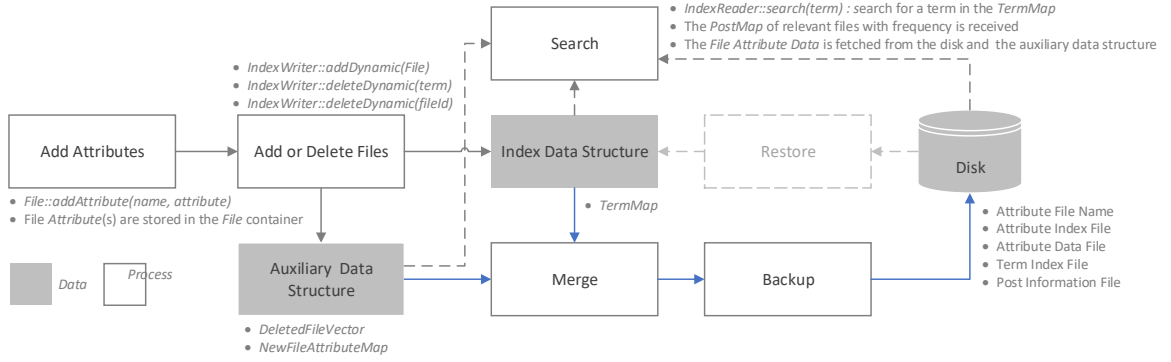


Figure A.10: Illustrates a dynamic index management process. This allows querying while at the same time deletion and addition of files to the index.

querying. After the new index is ready the old index is replaced. This solution does not scale well when users use the system regularly and in the meantime filesystem information changes frequently. In such situation a dynamic index management is essential to keep the index up-to-date and in the mean time provide querying facility. Methods to address such challenges are discussed in [136,198].

Here, we attempt to describe a simple dynamic index management technique that allows file addition, deletion and modification while at the same time facilitate querying. Figure A.10 illustrates the dynamic index creation, search, and back process. In this technique, we maintain two indexes:

- The existing **Inverted Index** and the **File Attribute Data** are accessed from the disk.
- A small auxiliary index is created to store the incoming (new and deleted) file information. The new files and their attribute data that need to be added to the existing index are stored in the `NewFileAttributeMap` data structure. The deleted file information that need to be updated to the existing index are stored in the `DeletedFileVector` list.
- Searches are run across both indexes and merged.
- When the size of the auxiliary index increases beyond certain limit, a merge operation updates the old index with the auxiliary index information.

The function `IndexWriter::addDynamic(File)` (A.4) adds the new file and its attribute information in in the `NewFileAttributeMap`. When a document is updated or deleted a separate delete document vector (`DeletedFileVector`)

maintains that information to make sure that the stale data is not read. The deleted file and term information are updated by the functions A.6 and A.5.

Algorithm A.4 IndexWriter::addDynamic(File)

Input: File

1: IndexWriter::addFile(File) ▷ write to TermMap and NewFileAttributeMap

Algorithm A.5 IndexWriter::deleteDynamic(fileId)

Input: File

```
1: define PostMap <fileId, frequency>
2: input TermMap <term, PostMap>
3: input FileAttributeMap <fileId, Attribute>
4: for iterate TermMap do
5:   PostMap ← TermMap.second
6:   if PostMap.find(fileId) then
7:     PostMap.erase(fileId)
8:   end if
9: end for
10: if FileAttributeMap.find(fileId) then
11:   FileAttributeMap.erase(fileId)
12: end if
13: update DeletedFileVector
```

Algorithm A.6 IndexWriter::deleteDynamic(term)

Input: term

```

1: define FileVector
2: define AllFileVector
3: define DeletedFileVector
    ▷ Find the documents associated to the input term.
4: found ← TermMap.find(term)
5: if found then
6:   PostMap ← found.second
7:   for iterate PostMap do
8:     FileVector ← PostMap.first
9:   end for
10: else
11:   return false
12: end if
    ▷ Erase the term from the index.
13: TermMap.erase(term)
    ▷ Store all the document ids in a vector.
14: for iterate TermMap do
15:   PostMap ← TermMap.second
16:   for iterate PostMap do
17:     AllFileVector ← PostMap.first
18:   end for
19: end for
    ▷ Files to be deleted: present in FileVector but not in AllFileVector.
20: for  $i \leftarrow 0, \text{FileVector.size()} - 1$  do
21:   fileId ← FileVector[i]
22:   found ← find(AllFileVector.start(), AllFileVector.end(), fileId)
23:   if not found then
24:     DeletedFileVector ← fileId
25:   end if
26: end for
    ▷ Delete all the fileIds contained in DeletedFileVector from the
    FileAttributeMap.
27: for  $i \leftarrow 0, \text{DeletedFileVector.size()} - 1$  do
28:   fileId ← DeletedFileVector[i]
29:   if FileAttributeMap.find(fileId) then
30:     FileAttributeMap.erase(fileId)
31:   end if
32: end for
33: update DeletedFileVector    ▷ delete from TermMap and FileAttributeMap

```

Appendix B

Stimuli For Empirical Studies

Contents

B.1 Stimuli for Experiments One, Two and Three	165
B.2 Focus Group Study: Questionnaire	165
B.3 Results of The Questionnaire	171

This appendix contains additional material related to the Chapter 5. We present (a) the stimuli used in the empirical studies and (b) the questionnaire and the results of the focus group study.

B.1 Stimuli for Experiments One, Two and Three

Example screenshots of Experiment One, Two, and Three that were showed to the participants during the empirical study are presented in Figure B.1, B.3, and B.5 respectively. The stimuli used in Experiment One, Two, and Three are illustrated in Figure B.2, B.4, and B.6 respectively.

B.2 Focus Group Study: Questionnaire

The questionnaire asked to the focus group during the post-development evaluation is divided into four parts (a) static dots vs. animated dots, (b) static dots vs. static glyphs, (c) static glyphs vs. animated glyphs, and (d) animated dots vs. animated glyphs.

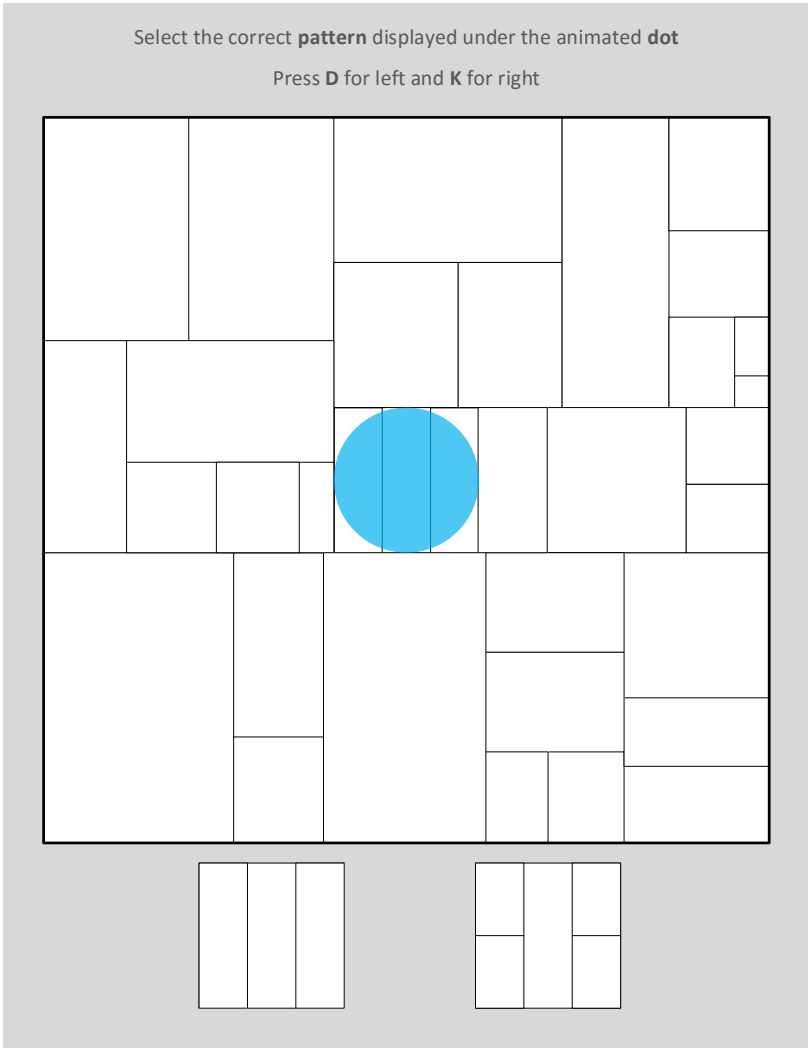
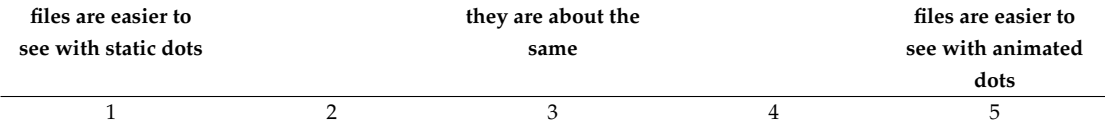


Figure B.1: Example screenshots of Experiment One as presented to the participants.

B.2.1 Static dots vs. animated dots

Q1a. Do you find it is easier to see **all individual files** (identified by a search) when they are presented as static dots or animated dots?



Q1b. Do you find it is easier to see the **underlying directory (folder) structure** in a treemap when files are shown as static dots or animated dots?

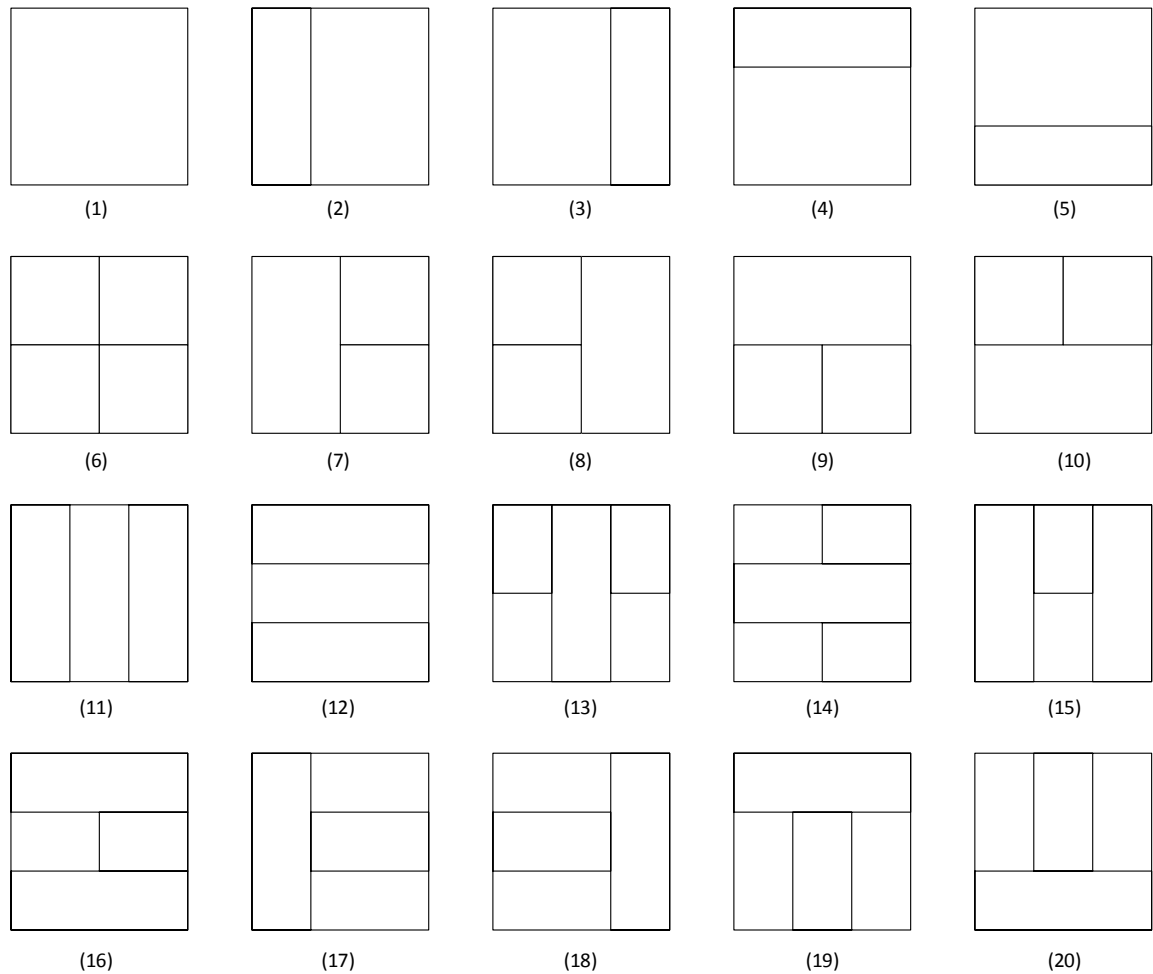
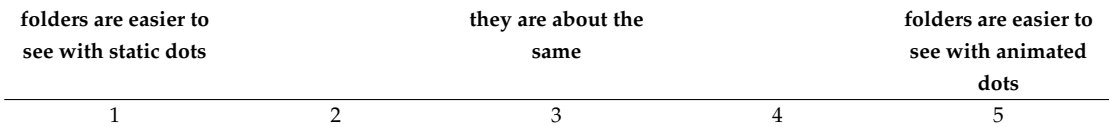
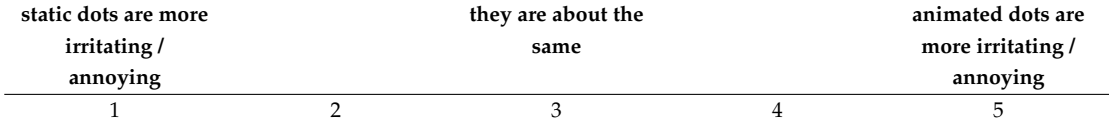


Figure B.2: Stimuli (patterns) used in Experiment One.



Q1c. Do you find static dots or animated dots more **irritating/annoying**?



B.2.2 Static dots vs. static glyphs

Q2a. How easy or difficult is for you to **learn** the visual encoding of file attributes in glyphs?

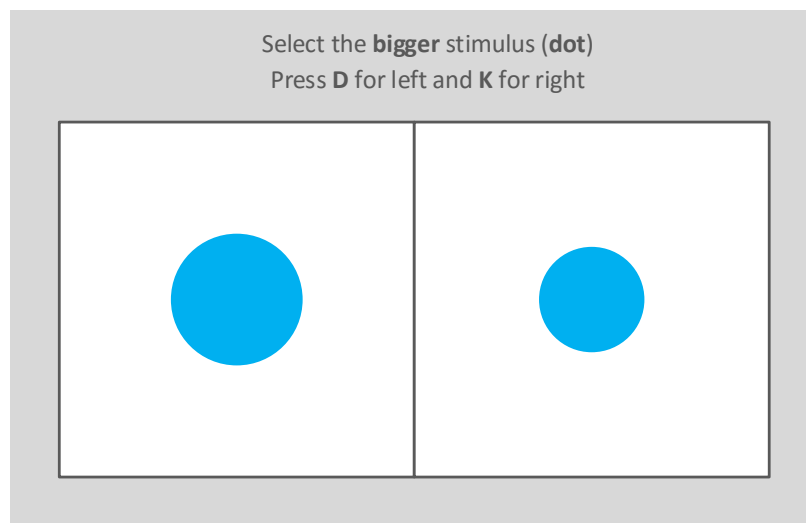


Figure B.3: Example screenshots of Experiment Two as presented to the participants.

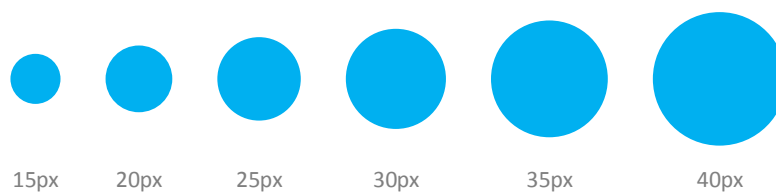


Figure B.4: Stimuli used in Experiment Two.

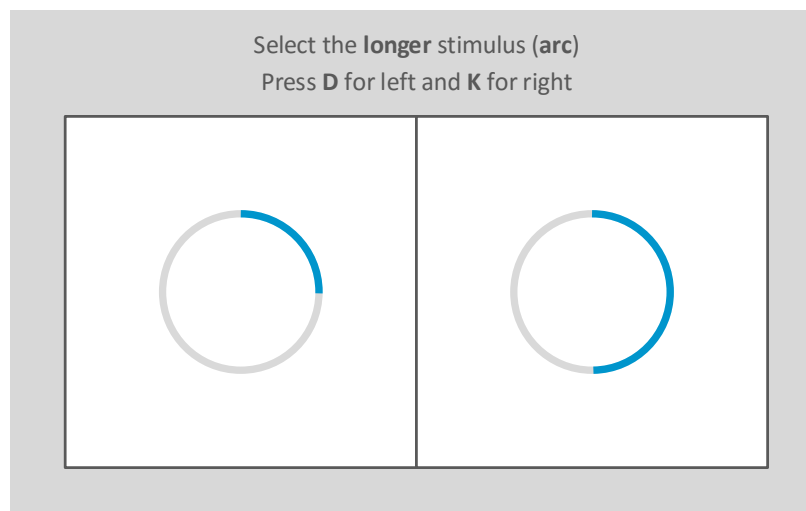
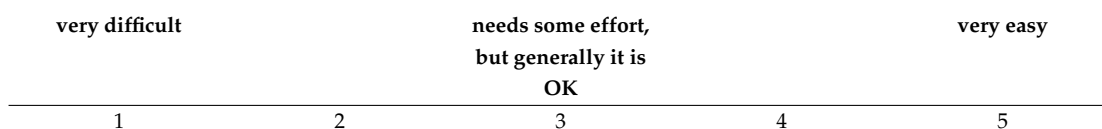


Figure B.5: Example screenshots of Experiment Three as presented to the participants.



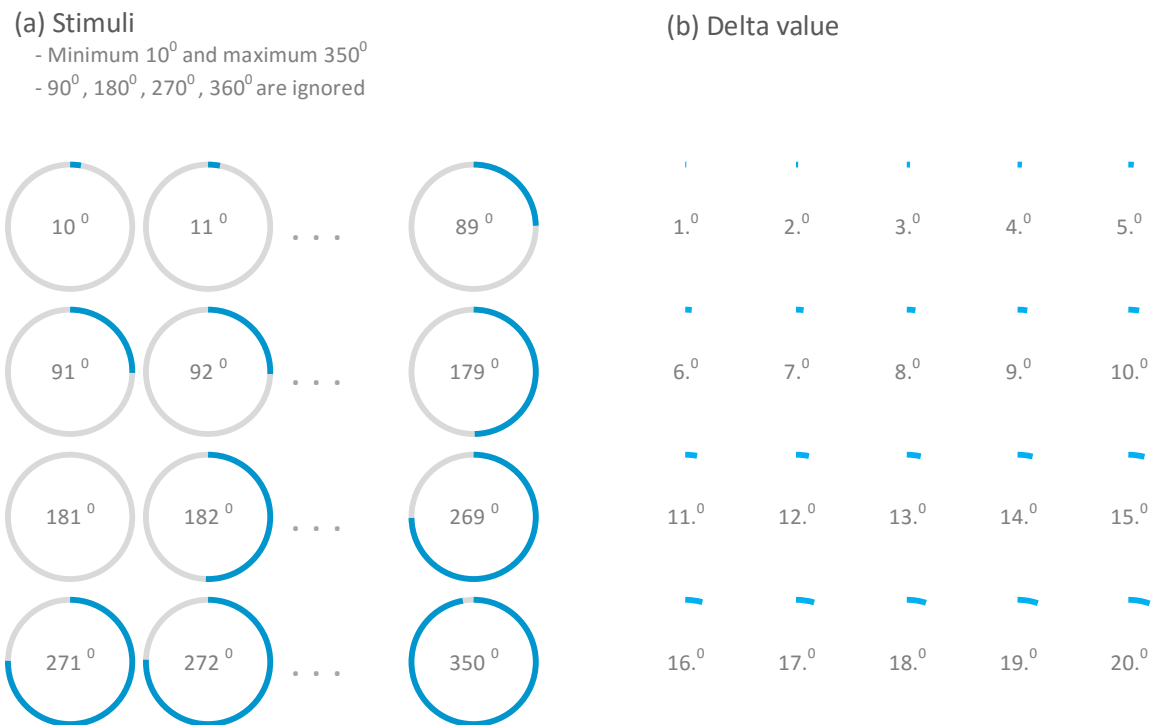
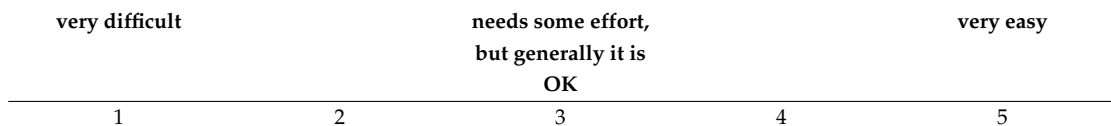
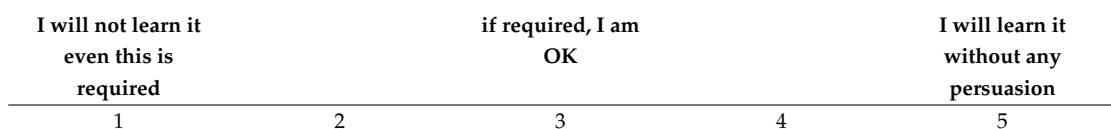


Figure B.6: Stimuli used in Experiment Three.

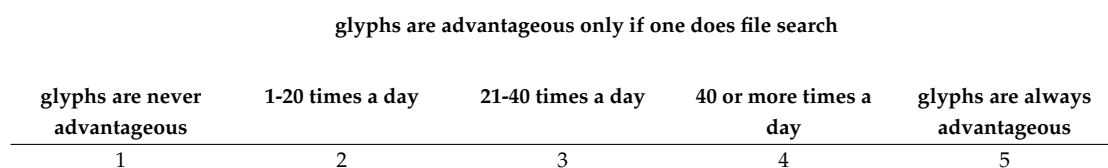
Q2b. How easy or difficult is for you to **remember** the visual encoding of file attributes in glyphs?



Q2c. How willing are you to **learn and memorise** the visual encoding of file attributes in glyphs if your job requires you to carry out file search frequently (e.g. more than 20 times per day)?



Q2d. In order for the glyphs to have advantages over static dots, how frequently do you feel that one will have to carry out file searches as part of one's job?



Q2e. Do you find it is easier to **see the underlying directory (folder) structure** in a treemap when files are shown as static dots or static glyphs?

folders are easier to see with static dots		they are about the same		folders are easier to see with static glyphs
1	2	3	4	5

B.2.3 Static glyphs vs. animated glyphs

Q3a. Do you find it is easier to **see all individual files** (identified by a search) when they are presented as static glyphs or animated glyphs?

files are easier to see with static glyphs		they are about the same		files are easier to see with animated glyphs
1	2	3	4	5

Q3b. Do you find it is easier to **see individual file attributes** (e.g., type, size, last modified date) when they are presented as static glyphs or animated glyphs?

attributes are easier to see with static glyphs		they are about the same		attributes are easier to see with animated glyphs
1	2	3	4	5

Q3c. Do you find it is easier to **see the underlying directory (folder) structure** in treemap when files are shown as static glyphs or animated glyphs?

folders are easier to see with static glyphs		they are about the same		folders are easier to see with animated glyphs
1	2	3	4	5

Q3d. Do you find static glyphs or animated glyphs more **irritating / annoying**?

static glyphs are more irritating / annoying		they are about the same		animated glyphs are more irritating / annoying
1	2	3	4	5

B.2.4 Animated dots vs. animated glyphs

Q4a. Do you find it is easier to **see the underlying directory (folder) structure** in a treemap when files are shown as animated dots or animated glyphs?

folders are easier to see with animated dots	they are about the same			folders are easier to see with animated glyphs
1	2	3	4	5

Q4b. Do you find animated dots or animated glyphs more **irritating / annoying**?

animated dots are more irritating / annoying	they are about the same			animated glyphs are more irritating / annoying
1	2	3	4	5

B.3 Results of The Questionnaire

In this section we present the results of the questionnaire as described in B.2.

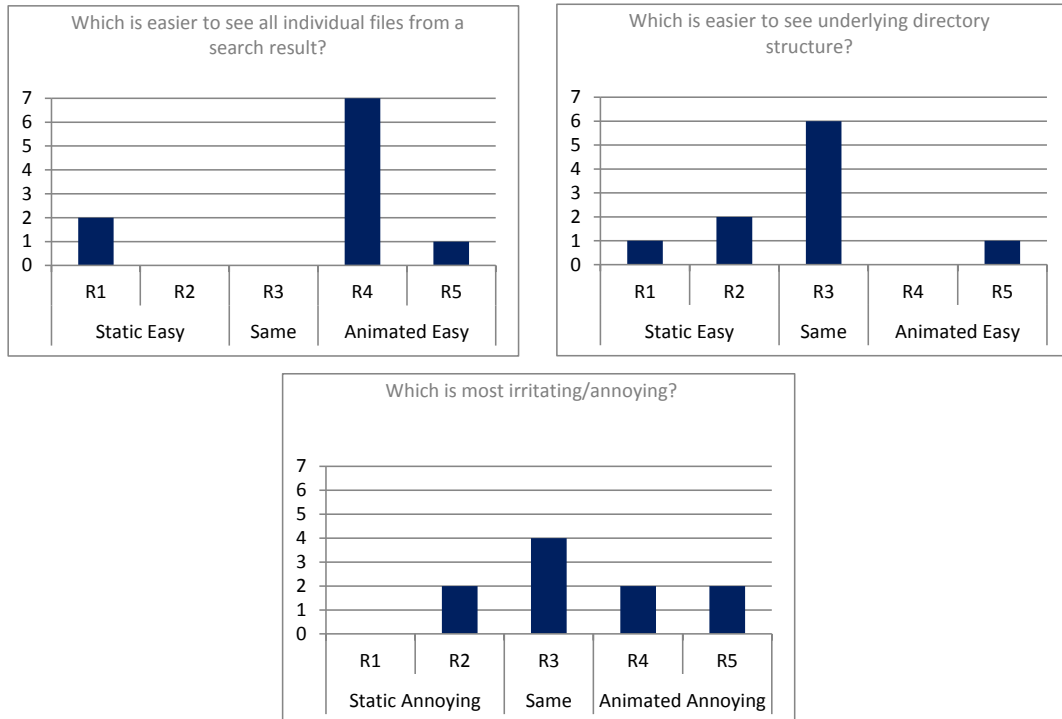


Figure B.7: Results of the static dots vs. animated dots.

APPENDIX B. STIMULI FOR EMPIRICAL STUDIES

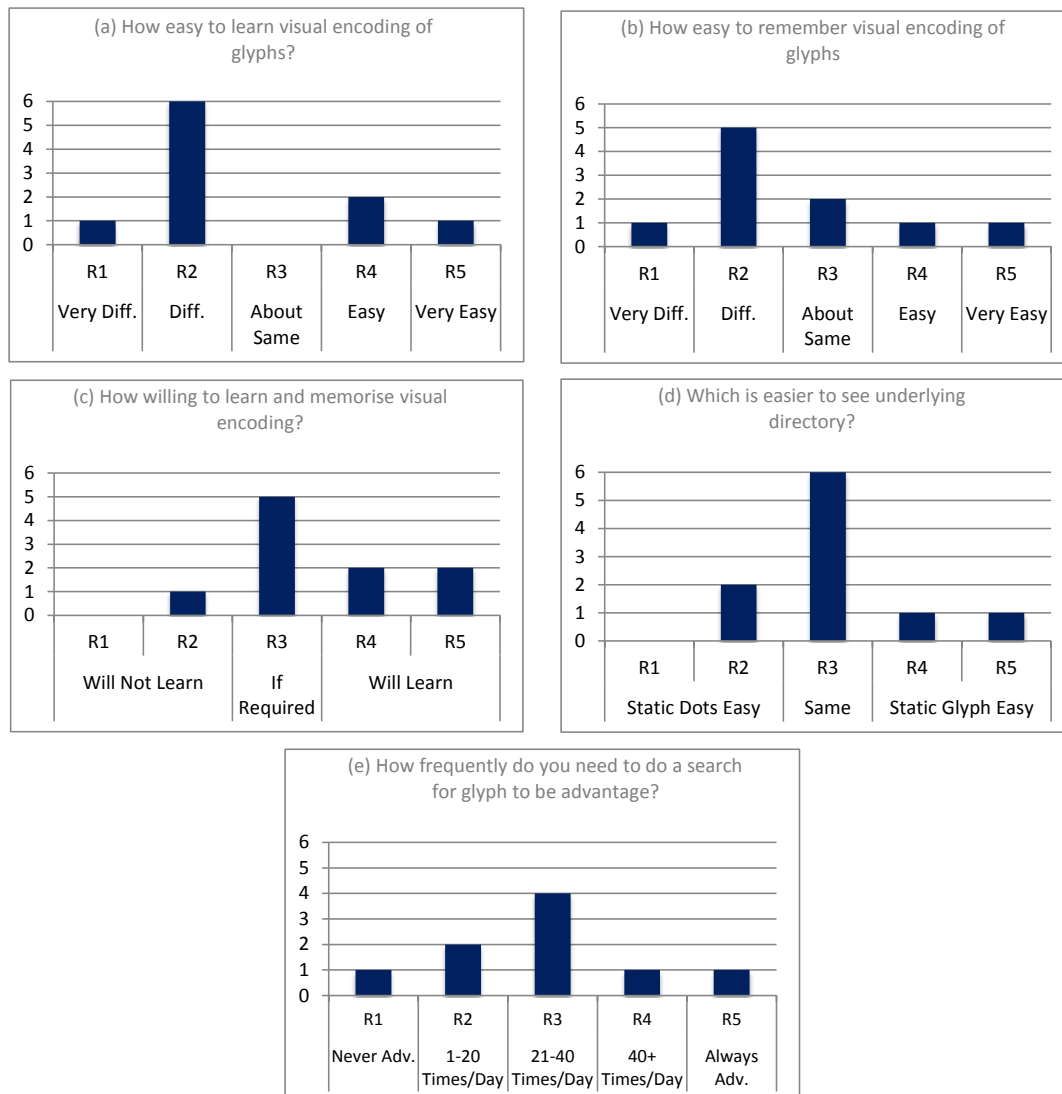


Figure B.8: Results of the static dots vs. static glyphs.

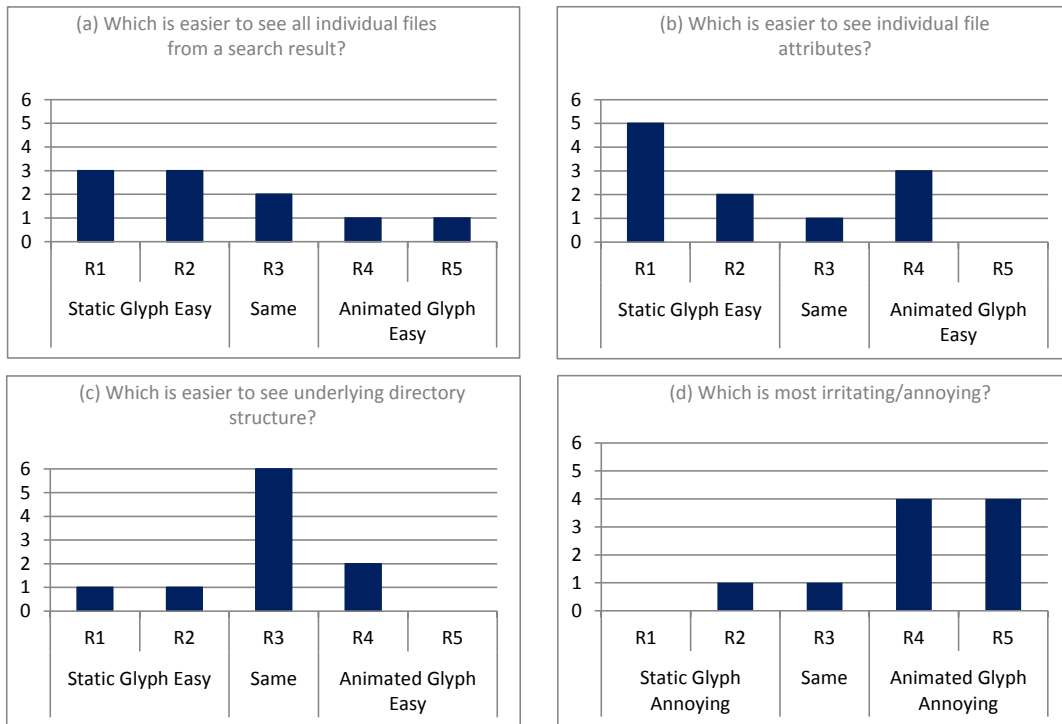


Figure B.9: Results of the static glyphs vs. animated glyphs.

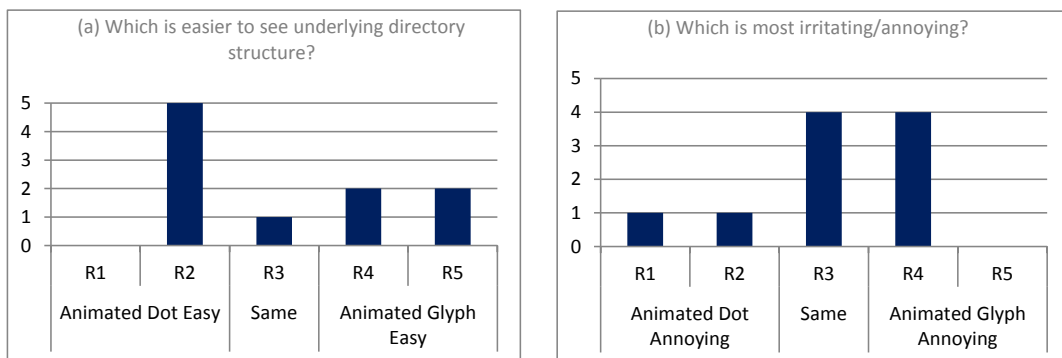


Figure B.10: Results of the animated dots vs. animated glyphs.

List of Figures

- 2.1 An example relational data model. Person and companies details are stored in a Person and a Company table. A WorksIn table contains the relationships between person data and company data. . . . 11
- 2.2 An example document data model. Instead of splitting objects into tables and rows, they are represented in two different document structures one for persons and one for a company. By providing a reference in the person document to a company document, we have created a relationship between the two entities. 12
- 2.3 An example of a graph data model. Here, each instance of people and company data are represented as nodes, and edges between the nodes represent the relationships between the nodes. As many attributes as required could be added to both edges and nodes. A directed edge has an arrow pointing to its head node while an undirected edge does not have any arrows. 13
- 2.4 (a) HotMap: the term-frequency (tf) of each keyword in the query is shown using colour coding. (b) Concept Highlighter: Firstly a set of concepts related to the search query is displayed, and based on the user's selection the search results are sorted. The sorting is based on a fuzzy membership score and highlighted using colour coding. Adopted from [105]. 19
- 2.5 Gradual refinement of query parameters in VisGets using the dimension: (a) spatial– the region close to Caribbean is selected, (b) topical– the larger tag “politics” is selected, (c) and (d) temporal– first the month February is selected and then few days around the middle of February are brushed. Adopted from [66]. 20

2.6	Fluid Views: (a) Zooming increases geographical detail in the map, here, from a continental view to the Mediterranean. (b) The time scale is changed from months, to weeks, and days. (c) Base layer gets gradually magnified and the items, filtered and displayed with more detail. Adopted from [67].	20
2.7	PivotPaths tool shows variety of relation types across various dimensions. Adopted from [68].	21
2.8	Search results are displayed with Voronoi treemap. Adopted from [153].	22
2.9	ProjSnippet tool visualizes the Web search result and clusters them based on content similarity. Adopted from [91].	22
2.10	Footprints, a document search and visualization tool. Adopted from [109].	22
2.11	“/opt” directory of an example GNU/Linux system and its contents are visualized with (a) File System Explorer (b) xdiskusage, and (c) Disk Usage Analyzer.	23
2.12	Screenshot of k4dirstat tool visualizing the “opt” directory (was also visualized in Figure 2.11) of a GNU/Linux file system using a treemap.	25
2.13	Different treemaps with non-rectangular layout: (a) Jigsaw [218], (b) Convex [156], (c) Voronoi [15], created from <code>cse512-14w.github.io/fp-plvines-djpeter/demo.html</code> , (d) Circular <code>lip.sourceforge.net/ctreemap.html</code> , (e) Gosper [10].	27
2.14	Pixel-based visualization, right image without visual boosting and left image shows visual boosting with background colouring and semi-transparent halos. Adopted from [154].	28
2.15	Multivariate information visualization techniques: tables (left), scatter plots (middle), and parallel coordinate plots (right). Adopted from [149].	28
2.16	(a) Simple glyph depicting wind direction, speed, temperature, and location. (b) A complex glyph by Duffy <i>et al.</i> [72] encoding 23 attributes of sperm cells (adopted from [72]).	29
2.17	Example text-based provenance view, which shows different attributes and relationships between different documents. Adopted from [107].	33

2.18	This example node-link visualization shows the work done in the area of operating systems, workflow, and database provenance visualization from 1991 to 2013. The nodes shown as dots and the links shown as lines connecting the nodes. Adopted from [37].	34
2.19	VisTrail: (a) node-link view of the dataflow, (b) spreadsheet view shows different visualizations of the data instances. Adopted from [36].	34
2.20	Workflow provenance information recorded in text using glyph-based graph visualization. Each node is enriched with multivariate glyphs to visualize more information. Adopted from [135].	35
2.21	The InProv tool was developed to visualize file system provenance. The left image shows the interactions of the “bash” node with its parent nodes (connected by blue edges) and child nodes (connected by red edges). The right image shows “node stack” and “context views” to provide browsing history and location context. Adopted from [29].	36
3.1	The three different strategies considered are illustrated in this figure. (1) In <i>Database</i> the task may include capture and storage in a file system, to enter files into a database, search in the database, and finally package and delivery. (2) In <i>Enterprise Search Engine</i> the task of recording each file in a database is eliminated and here the file search over a distributed file system is through an enterprise search engine. (3) In <i>Visualization-assisted Enterprise Search Engine</i> the search using the enterprise search engine is supported by visualization capabilities.	43
3.2	The modules of the enterprise search engine.	48

3.3	Top window: an example screenshot of the search interface, which is designed to be used with two computer displays. The components shown in the first display are (1) the search query form, (2) the tabular view of search results, (3) the graph view of SPG illustrating how the user (focus) collaborated with others, (4) the collaboration summary view, illustrates the similarity between the search queries performed by the administrators, (5) glyph-based view, uses multivariate glyphs for search queries and search results which are shown above and below the central line respectively, and (6) are the dialogs for showing the queries and the late-discovered files. Bottom window: the component shown in the second display is a treemap visualization of a file system and the search results encoded in multivariate glyphs.	49
3.4	A flowchart of the VAESE system illustrating a typical search process and the visualization of the search results and provenance. . . .	50
4.1	The architecture of the enterprise search engine. This chapter focuses on the <i>Search</i> part: multiple <i>Ontology</i> assisted (a) query expansion, (b) field-based ranking, and (c) active learning of the ontologies. The search engine also facilitates the functionalities (e.g., the search space information and ontologies for computing semantic query similarity in SPG) for the integration of visualization components to be discussed in following chapters.	68
4.2	A snippet of the term ontology shows the weights of the search term “building” and its related terms in one-link away. In this example, four different sets of users feedback data are used to train the ontologies, showing the RFIs and changes in weights before (w') and after normalization (wn).	81
4.3	Comparison of the precision and recall before and after training with three different training sets.	85
4.4	The CDF of query processing time, 500 queries over 0.5 million files.	86
4.5	(a) The mean and median of execution times for queries vs. dataset size. (b) The index size vs. dataset size.	88
5.1	The conventional depiction of search results as coloured-dots on a treemap results in great difficulties in visual search. In the background/context green are directories and purple are files.	95

5.2	When multivariate glyphs are used the occlusion would make the underlying context difficult to see.	95
5.3	We propose to use cyclically animated glyphs to address both difficulties shown in Figure 5.1 and 5.2.	96
5.4	File attribute glyph. Four different parameters (type, size, date and time) are encoded into the glyph.	101
5.5	Visualization of file search, featuring glyphs for search results (focus) and a treemap for the file system (context).	104
5.6	Design options of temporal transfer functions. Each TTF is shown as a time series, with x -axis shows the normalized time, and y -axis shows the switch between focus (1) and context (0). The five TTFs in each row represent different focus/context ratio, while the three rows represent different transition periods from minimum to maximum. The values of transition periods are 0, 300, and 375 ms.	105
5.7	Results of the Controlled Experiment One.	108
5.8	Results of the Controlled Experiment Two (D5: 5 pixels difference, D15: 15 pixels difference, and so on).	110
5.9	Results of the Controlled Experiment Three.	111
6.1	In a traditional query-flow graph, each node represents a query and the weight $w_{i,j}$ of the directed edge from Q_i to Q_j represents the likelihood that the two queries are part of the same search session.	123
6.2	In the overall SPG, there are different types of nodes and edges. The nodes are hierarchically organized.	124
6.3	Glyph design for search provenance graphs. (a) Six different design options for characterizing a query, (b) the colours and icons used to highlight different query fields, and (c) the glyph components to depict a file in search results (FP: False Positive).	130
6.4	This example SPG graph view of SPG displayed on Julie's search provenance window. It illustrates how Julie (focus) collaborated with other administrators. For example, Julie's fourth query is formulated based on Charlotte and in Julie's second day of search she formulated two queries based on Andy. The last nine queries from Julie's search are selected for glyph-based view as shown in Figure 6.5.	134

- 6.5 An example glyph-based visualization of the queries selected in Figure 6.4. The search queries are shown above the central line, and the search results are shown beneath the central line. Each search query may feature four criteria, clockwise, keyword, type, size and date. An icon appears when this criterion is new or has been changed. The search results are shown in two forms: summary statistics are shown when research results consist of more than 25 files and macro-glyph otherwise. For search results, green represents newly found files, gray represent files common to the current and previous queries, blue represents files appeared only in the previous query, and the red represents the selected files. The background colour of the 'Late Discovery' dialog is gray as there is currently no newly discovered file. 135
- 6.6 This example collaboration summary view of SPG illustrates the similarity between the search queries performed by the administrators associated with a project. 137
- A.1 An index creation and search process. Here, searches happen in the *Inverted Index* (`TermMap`) and the *File Attribute Data* (`FileAttributeMap`) stored in memory. 153
- A.2 (a) A `file Attribute` structure holds all file attribute names, e.g., file name, type, size etc. (b) `AttributeMap` holds all the attribute name and value of a file as a key-value pair. (c) In this example, a file `/home/budget/budget.xlsx` of 1024 byte size is indexed in a `AttributeMap` container. 153
- A.3 (a) *File Attribute Data*— each file is assigned with a unique identifier (`fileId`), and its attribute values are stored in a `FileAttributeMap`. (b) In this example, two files `/home/budget/budget.xlsx` and `/home/budget/finance.docx` are assigned with `fileId`: 0 and 1 respectively. 154
- A.4 (a) *Inverted index*— is stored in the `TermMap`. It contains the term (as key) and the files it is appearing on with its frequency in the post map (`PostMap`). (b) The example inverted index is created from files illustrated in Figure A.3 (b). 154
- A.5 The format of *Attribute Name File* (ANF). The `Attribute` data structure is saved in this file. 157

A.6	(a) The formats of Attribute Index File (AIF) and Attribute Data File (ADF). These files stores the <code>FileAttributeMap</code> . (b) This example shows a AIF and a ADF storing the <code>FileAttributeMap</code> illustrated in Figure A.3 (b).	157
A.7	(a) The formats of Term Index File (TIF) and Post Information File (PIF). The inverted index information, i.e., the <code>TermMap</code> is saved in the disk using these two files. (b) This example shows a TIF and a PIF storing the <code>TermMap</code> illustrated in Figure A.4 (b).	158
A.8	Index management using in-memory and disk data. Here, a part of inverted index (<code>TermMap</code>) is loaded in memory. The keywords are searched in the <code>TermMap</code> and the corresponding file attributes are accessed from the PIF, AIF, and ADF stored in the disk.	160
A.9	This example illustrates a search to retrieve all the files related to the keyword "home". The inverted index (the <code>TermMap</code>) is accessed from memory and the file attribute data is accessed from the files stored in the disk.	160
A.10	Illustrates a dynamic index management process. This allows querying while at the same time deletion and addition of files to the index.	161
B.1	Example screenshots of Experiment One as presented to the participants.	166
B.2	Stimuli (patterns) used in Experiment One.	167
B.3	Example screenshots of Experiment Two as presented to the participants.	168
B.4	Stimuli used in Experiment Two.	168
B.5	Example screenshots of Experiment Three as presented to the participants.	168
B.6	Stimuli used in Experiment Three.	169
B.7	Results of the static dots vs. animated dots.	171
B.8	Results of the static dots vs. static glyphs.	172
B.9	Results of the static glyphs vs. animated glyphs.	173
B.10	Results of the animated dots vs. animated glyphs.	173

List of Tables

2.1	Key features of commercial BIM software (to continue in Table 2.2). .	37
2.2	Continued from Table 2.1.	38
3.1	Statistics of three Laing O'Rourke Projects– files captured and stored in distributed file system, files recorded in RDBMS, and the number of files issued to the customer. [Source: Laing O'Rourke]	58
3.2	Statistics for life-cycle of files costs in person-days (8 person-hours per person-day). [Source: Laing O'Rourke]	60
3.3	Search-only cost in person-days (8 person-hours per person-day): local search in database, centralized search using enterprise search engine, and centralized search using visualization-assisted enterprise search engine. [Source: Laing O'Rourke]	60
4.1	Size distribution for different types of files	82

List of Algorithms

4.1	NormalizeWeights (w'_i)	80
A.1	File::addAttribute(name, attribute)	156
A.2	IndexWriter::addFile(File)	156
A.3	IndexReader::search(term)	159
A.4	IndexWriter::addDynamic(File)	162
A.5	IndexWriter::deleteDynamic(fileId)	162
A.6	IndexWriter::deleteDynamic(term)	163

Bibliography

- [1] A. Abdul-Rahman, J. Lein, K. Coles, E. Maguire, M. Meyer, M. Wynne, C. R. Johnson, A. Trefethen, and M. Chen. Rule-based Visual Mappings, with a Case Study on Poetry Visualization. *Comp. Graphics Forum*, 32(3), 2013. 30, 97
- [2] A. Abdul-Rahman, E. Maguire, and M. Chen. Comparing Three Designs of Macro-Glyphs for Poetry Visualization. In *Eurographics Conf. on Visualization*, 2014. 31, 131
- [3] R. A. Abrams and S. E. Christ. Motion onset captures attention. *Psychological Science*, 14(5):427–432, 2003. 93
- [4] N. Agrawal, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Generating Realistic Impressions for File-system Benchmarking. In *Proc. 7th Conf. on File and Storage Technologies*, pages 125–138, 2009. 83, 84
- [5] S. Allard, K. J. Levine, and C. Tenopir. Design engineers and technical professionals at work: Observing information usage in the workplace. *Journal of the American Society for Information Science and Technology*, 60(3):443–454, mar 2009. 2
- [6] Y. Amsterdamer, S. B. Davidson, D. Deutch, T. Milo, J. Stoyanovich, and V. Tannen. Putting lipstick on pig: enabling database-style workflow provenance. In *Proc. VLDB Endowment*, volume 5, pages 346–357, 2011. 35
- [7] K. Andrews, V. Sabol, W. Lackner, C. Gütl, and J. Moser. Search Result Visualisation with xFIND. In *Proc. 2nd Int. Workshop on User Interfaces to Data Intensive Systems*, page 50, 2001. 19, 94
- [8] C. Andronis, A. Sharma, V. Virvilis, S. Deftereos, and A. Persidis. Literature mining, ontologies and information visualization for drug repurposing. *Briefings in bioinformatics*, 12(4):357–68, 2011. 123

- [9] R. Angles. A comparison of current graph database models. In *IEEE 28th Int. Conf. on Data Engineering Workshops*, pages 171–177, 2012. 71
- [10] D. Auber, C. Huet, A. Lambert, B. Renoust, A. Sallaberry, and A. Saulnier. GosperMap: using a Gosper Curve for laying out hierarchical data. *IEEE Trans. on Visualization & Comp. Graphics*, 19(11):1820–32, 2013. 26, 27, 176
- [11] A. D. Baddeley. Short-term memory for word sequences as a function of acoustic, semantic and formal similarity. *Quarterly Journal of Experimental Psychology*, 18(4):362–365, 1966. 93
- [12] R. Baeza-Yates. Graphs from Search Engine Queries. In *Proc. 33rd Conf. on Current Trends in Theory and Practice of Comp. Sc.*, volume 4362 of *LNCS*, pages 1–8, 2007. 123
- [13] P. Bak, F. Mansmann, H. Janetzko, and D. A. Keim. Spatiotemporal analysis of sensor logs using growth ring maps. *IEEE Trans. on Visualization & Comp. Graphics*, 15(6):913–920, 2009. 97
- [14] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *Proc. SIGIR*, page 43, 2006. 1
- [15] M. Balzer, O. Deussen, and C. Lewerentz. Voronoi Treemaps for the Visualization of Software Metrics. In *Proc. ACM Symp. on Software Visualization*, page 165, 2005. 26, 27, 176
- [16] L. Bavoil, S. P. Callahan, C. E. Scheidegger, P. J. Crossno, C. T. Silva, J. Freire, and H. T. Vo. VisTrails: Enabling Interactive Multiple-View Visualizations. In *IEEE VIS*, pages 135–142, 2005. 35, 122
- [17] O. Ben-Yitzhak, S. Yogev, N. Golbandi, N. Har’El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. Shekita, and B. Sznajder. Beyond basic faceted search. In *Proc. WSDM*, page 33, 2008. 18
- [18] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983. 29, 97
- [19] J. Bhogal, A. MacFarlane, and P. Smith. A review of ontology based query expansion. *Information Processing & Management*, 43(4):866–886, 2007. 18, 39, 67

-
- [20] A. Billig, E. Blomqvist, and F. Lin. Semantic Matching Based on Enterprise Ontologies. In *On the Move to Meaningful Internet Systems*, volume 4803 of *LNCIS*, pages 1161–1168. 2007. 17, 66
- [21] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2007. 78
- [22] O. Biton, S. Cohen-Boulakia, and S. B. Davidson. Zoom*UserViews: querying relevant provenance in workflow systems. In *Proc. VLDB*, pages 1366–1369, 2007. 35
- [23] O. Biton, S. Cohen-Boulakia, S. B. Davidson, and C. S. Hara. Querying and Managing Provenance through User Views in Scientific Workflows. In *IEEE 24th Int. Conf. on Data Engineering*, pages 1072–1081, 2008. 35, 122
- [24] R. Blanch and E. Lecolinet. Browsing Zoomable Treemaps: Structure-Aware Multi-Scale Navigation Techniques. *IEEE Trans. on Visualization & Comp. Graphics*, 13(6):1248–1253, 2007. 25
- [25] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The Query-flow Graph: Model and Applications. In *Proc. CIKM*, pages 609–618, 2008. 124, 145
- [26] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. Query reformulation mining: models, patterns, and applications. *Information Retrieval*, 14(3):257–289, 2010. 126, 145
- [27] I. Bordino, C. Castillo, D. Donato, and A. Gionis. Query similarity by projecting the query-flow graph. In *Proc. SIGIR*, page 515, 2010. 126, 145
- [28] R. Borgo, J. Kehler, D. H. S. Chung, E. Maguire, R. S. Laramée, H. Hauser, M. Ward, and M. Chen. Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications. *Eurographics State of the Art Reports*, pages 39–63, 2013. 28, 29, 30
- [29] M. A. Borkin, C. S. Yeh, M. Boyd, P. Macko, K. Z. Gajos, M. Seltzer, and H. Pfister. Evaluation of filesystem provenance visualization tools. *IEEE Trans. on Visualization & Comp. Graphics*, 19(12):2476–85, dec 2013. 35, 36, 122, 177
- [30] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics*, 17(12):2301–2309, 2011. 48

- [31] R. P. Botchen, S. Bachthaler, F. Schick, M. Chen, G. Mori, D. Weiskopf, and T. Ertl. Action-based multifield video visualization. *IEEE Trans. on Visualization & Comp. Graphics*, 14(4):885–899, 2008. 31
- [32] B. Bridgeman, A. H. C. der Heijden, and B. M. Velichkovsky. A theory of visual stability across saccadic eye movements. *Behavioral and Brain Sciences*, 17(02):247–258, 1994. 92
- [33] M. Bruls, K. Huizing, and J. J. Van Wijk. Squarified Treemaps. In *In Proc. Joint Eurographics and IEEE TCVG Symp. on Visualization*, pages 33–42, 1999. 103
- [34] A. Buccella, A. Cechich, and P. Fillottrani. Ontology-driven geographic information integration: A survey of current approaches. *Comp. & Geosciences*, 35(4):710–723, 2009. 71
- [35] Y. Cai, X. L. Dong, A. Halevy, J. M. Liu, and J. Madhavan. Personal information management with SEMEX. In *Proc. SIGMOD*, pages 921–923, 2005. 14, 66
- [36] S. P. Callahan, J. Freire, E. Santos, C. E. Scheideggers, C. T. Silva, and H. T. Vo. VisTrails: Visualization Meets Data Management. In *Proc. SIGMOD*, page 745, jun 2006. 34, 35, 46, 119, 122, 177
- [37] L. Carata, S. Akoush, N. Balakrishnan, T. Bytheway, R. Sohan, M. Selter, and A. Hopper. A primer on provenance. *Communications of the ACM*, 57(5):52–60, may 2014. 33, 34, 119, 122, 177
- [38] S. K. Card, J. D. Mackinlay, and B. Schneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999. 98
- [39] S. K. Card, P. Pirolli, M. Van Der Wege, J. B. Morrison, R. W. Reeder, P. K. Schraedley, and J. Boshart. Information scent as a driver of web behavior graphs: results of a protocol analysis method for web usability. In *Proc. SIGCHI*, pages 498–505, mar 2001. 32, 33, 40, 122
- [40] S. Carpendale, M. Chen, D. Evanko, N. Gehlenborg, C. Gorg, L. Hunter, F. Rowland, M.-A. Storey, and H. Strobel. Ontologies in biological data visualization. *IEEE Comp. Graphics and Applications*, 34(2):8–15, 2014. 123

-
- [41] P. Castells, M. Fernandez, and D. Vallet. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. *IEEE Trans. on Knowledge and Data Engineering*, 19(2):261–272, 2007. 17, 66, 82, 146
- [42] M. Chau. Visualizing web search results using glyphs: Design and evaluation of a flower metaphor. *ACM Trans. on Management Information Systems*, 2(1):1–27, 2011. 19, 30, 96
- [43] C. Chen. CiteSpace-II: Detecting and visualizing emerging trends and transient patterns in scientific literature. *Journal of the American Society for Information Science and Technology*, 57(3):359–377, 2006. 30, 97
- [44] J. Chen, H. Guo, W. Wu, and C. Xie. Search your memory ! - an associative memory based desktop search system. In *Proc. SIGMOD*, pages 1099–1102, 2009. 17, 66
- [45] M. Chen, L. Floridi, and R. Borgo. What Is Visualization Really For? In *The Philosophy of Information Quality*, pages 75–93. Springer Synthese Library, 2014. 47
- [46] M. Chen and A. Golan. What May Visualization Processes Optimize? *IEEE Trans. on Visualization & Comp. Graphics*, (99):1–14, 2015. 47
- [47] P. Chen, B. Plale, Y.-W. Cheah, D. Ghoshal, S. Jensen, and Y. Luo. Visualization of network data provenance. In *19th Int. Conf. on High Performance Computing*, pages 1–9. IEEE, dec 2012. 34
- [48] H. Chernoff. The Use of Faces to Represent Points in K-Dimensional Space Graphically on JSTOR. *Journal of the American Statistical Association*, 68(342):361–368, 1973. 30
- [49] S. Chernov, G. Demartini, E. Herder, M. Kopycki, and W. Nejdl. Evaluating personal information management using an activity logs enriched desktop dataset. In *Proc. 3rd PIM Workshop*, 2008. 82
- [50] S. Chernov, P. Serdyukov, P.-A. Chirita, G. Demartini, and W. Nejdl. Building a Desktop Search Test-Bed. In *Advances in Information Retrieval*, volume 4425 of *LNCS*, pages 686–690. 2007. 82, 83
- [51] E. H. Chi. Improving Web usability through visualization. *IEEE Internet Computing*, 6(2):64–71, 2002. 32, 34, 40, 122

- [52] P.-A. Chirita, S. Costache, W. Nejdl, and R. Paiu. Beagle++ : Semantically Enhanced Searching and Ranking on the Desktop. In *The Semantic Web: Research and Applications*, volume 4011 of *LNCS*, pages 348–362. 2006. 17, 66
- [53] M. C. Chuah and S. G. Eick. Information rich glyphs for software management data. *IEEE Comp. Graphics & Applications*, 18(4):24–29, 1998. 31, 97
- [54] S. Cohen, C. Domshlak, and N. Zwerdling. On Ranking Techniques for Desktop Search. *ACM Trans. on Information Systems*, 26(2):1–24, 2008. 1, 15, 18, 39, 66, 68, 82
- [55] E. Cutrell, D. Robbins, S. Dumais, and R. Sarin. Fast, flexible filtering with phlat. In *Proc. SIGCHI*, pages 261–270, 2006. 15, 66
- [56] J. L. Dannemiller. Motion Popout in Selective Visual Orienting at 4.5 But Not at 2 Months in Human Infants. *Infancy*, 8(3):201–216, 2005. 93
- [57] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *Proc. SIGMOD*, page 1345, 2008. 35
- [58] N. Del Rio and P. P. Da Silva. Probe-it! Visualization Support for Provenance. In *Int. Conf. on Advances in Visual Computing*, pages 732–741, 2007. 34, 35, 122
- [59] T. Deng, L. Zhao, H. Wang, Q. Liu, and L. Feng. ReFinder: A Context-Based Information Refinding System. *IEEE Trans. on Knowledge and Data Engineering*, 25(9):2119–2132, 2013. 17, 67, 82
- [60] M. C. Díaz-Galiano, M. T. Martín-Valdivia, and L. A. Ureña-López. Query expansion with a medical ontology to improve a multimodal information retrieval system. *Computers in Biology and Medicine*, 39(4):396–403, 2009. 18, 67
- [61] M. D. Dikaiakos, A. Katsifodimos, and G. Pallis. Minersoft: Software Retrieval in Grid and Cloud Computing Infrastructures. *ACM Trans. on Internet Technology*, 12(1):1–34, 2012. 14, 66
- [62] J.-P. Dittrich and M. A. V. Salles. iDM: a unified and versatile data model for personal dataspace management. In *Proc. 32nd Int. Conf. on Very Large Data Bases*, pages 367–378, 2006. 15, 66

-
- [63] P. Dmitriev, P. Serdyukov, and S. Chernov. Enterprise and Desktop Search. In *WWW*, pages 1345–1346, 2010. 1, 18, 68
- [64] M. E. Dolan and J. A. Blake. Using ontology visualization to facilitate access to knowledge about human disease genes. *Applied Ontology*, 4(1):35–49, 2009. 123
- [65] A. Dong and H. Li. Multi-ontology based multimedia annotation for domain-specific information retrieval. In *IEEE Int. Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing*, volume 2, pages 158–165, 2006. 17, 67
- [66] M. Dörk, S. Carpendale, C. Collins, and C. Williamson. VisGets: coordinated visualizations for web-based information exploration and discovery. *IEEE Trans. on Visualization & Comp. Graphics*, 14(6):1205–12, 2008. 20, 21, 96, 119, 121, 175
- [67] M. Dörk, S. Carpendale, and C. Williamson. Fluid Views: A Zoomable Search Environment. In *Proc. Int. Working Conf. on Advanced Visual Interfaces*, page 233, 2012. 20, 21, 96, 121, 176
- [68] M. Dork, N. H. Riche, G. Ramos, and S. Dumais. PivotPaths: Strolling through Faceted Information Spaces. *IEEE Trans. on Visualization & Comp. Graphics*, 18(12):2709–2718, 2012. 21, 121, 176
- [69] M. Dörk, C. Williamson, and S. Carpendale. Towards Visual Web Search : Interactive Query Formulation and Search Result Visualization. *WWW Workshop on Web Search Result Summarization and Presentation*, pages 2–5, 2009. 21, 121
- [70] M. Dörk, C. Williamson, and S. Carpendale. Navigating Tomorrow’s Web: From Searching and Browsing to Visual Exploration. *ACM Trans. on the Web*, 6(3):1–28, 2012. 21, 121
- [71] G. M. Draper, Y. Livnat, and R. F. Riesenfeld. A survey of radial methods for information visualization. *IEEE Trans. on Visualization & Comp. Graphics*, 15(5):759–776, 2009. 97
- [72] B. Duffy, J. Thiyyagalingam, S. Walton, D. J. Smith, A. Trefethen, J. C. Kirkman-Brown, E. a. Gaffney, and M. Chen. Glyph-Based Video Visualization for Semen Analysis. *IEEE Trans. on Visualization & Comp. Graphics*, (99):1–14, 2013. 29, 31, 176

- [73] D. Duke, K. Brodlie, D. Duce, and I. Herman. Do You See What I Mean? *IEEE Comp. Graphics and Applications*, 25(3):6–9, 2005. 123
- [74] S. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I’ve seen: a system for personal information retrieval and re-use. In *Proc. SIGIR*, pages 72–79, 2003. 15, 32, 66, 122
- [75] C. Dunne, N. Henry Riche, B. Lee, R. Metoyer, and G. Robertson. GraphTrail: Analyzing Large Multivariate, Heterogeneous Networks while Supporting Exploration History. In *Proc. CHI*, page 1663, 2012. 34, 122
- [76] O. Eck and D. Schaefer. A semantic file system for integrated product data management. *Advanced Engineering Informatics*, 25(2):177–184, apr 2011. 16
- [77] B. El Idrissi, S. Baina, and K. Baina. Automatic generation of ontology from data models: A practical evaluation of existing approaches. In *IEEE 7th Int. Conf. on Research Challenges in Information Science*, pages 1–12, 2013. 18, 67
- [78] D. Elswailer and I. Ruthven. Towards task-based personal information management evaluations. In *Proc. SIGIR*, pages 23–30, 2007. 82
- [79] K. M. Evans and G. H. Kuenning. A Study of Irregularities in File-Size Distributions. In *Int. Symp. on Performance Evaluation of Comp. and Telecom. Systems*, 2002. 84
- [80] M. Farrugia and A. Quigley. Effective temporal graph layout: A comparative study of animation versus static display methods. *Information Visualization*, 10(1):47–64, 2011. 97
- [81] M. Fernández, I. Cantador, V. López, D. Vallet, P. Castells, and E. Motta. Semantically enhanced Information Retrieval: an ontology-based approach. *Web Semantics*, 9(4):434–452, 2011. 17, 67, 81
- [82] M. Fernandez, V. Lopez, M. Sabou, V. Uren, D. Vallet, E. Motta, and P. Castells. Using TREC for cross-comparison between classic IR and ontology-based search models at a Web scale. In *WWW*, 2009. 81
- [83] D. Fisher. Animation for visualization: opportunities and drawbacks. In J. Steele and N. Iliinsky, editors, *Beautiful Visualization*, chapter 19, pages 329–352. O’Reilly Media, 2010. 93

-
- [84] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani. Using Association Rules to Discover Search Engines Related Queries. In *Proc. 1st Conf. on Latin American Web Congress*, page 66. IEEE Comp. Society, 2003. 126
- [85] C. Friedrich and P. Eades. Graph Drawing in Motion. *Journal of Graph Algorithms & Applications*, 6(3):353–370, 2002. 106
- [86] G. Fuchs and H. Schumann. Visualizing abstract data on maps. In *Eighth Int. Conf. on Information Visualisation*, pages 139–144, 2004. 97
- [87] R. Fuchs and H. Hauser. Visualization of Multi-Variate Scientific Data. *Comp. Graphics Forum*, 28(6):1670–1690, 2009. 31, 97
- [88] G. Gao, Y.-S. Liu, M. Wang, M. Gu, and J.-H. Yong. A query expansion method for retrieving online BIM resources based on Industry Foundation Classes. *Automation in Construction*, 56:14–25, 2015. 37
- [89] A. Gehani and D. Tariq. SPADE: support for provenance auditing in distributed environments. In *Proc. 13th Int. Middleware Conf.*, pages 101–120, 2012. 34
- [90] O. Gilson, N. Silva, P. Grant, and M. Chen. From Web Data to Visualization via Ontology Mapping. *Comp. Graphics Forum*, 27(3):959–966, 2008. 123
- [91] E. Gomez-Nieto, F. San Roman, P. Pagliosa, W. Casaca, E. S. Helou, M. C. F. de Oliveira, and L. G. Nonato. Similarity preserving snippet-based visualization of web search results. *IEEE Trans. Visualization & Comp. Graphics*, 20(3):457–70, 2014. 21, 22, 121, 126, 145, 176
- [92] M. Gray, A. Badre, and M. Guzdial. Visualizing usability log data. In *Proc. IEEE Symp. on Information Visualization*, pages 93–98, 1996. 32, 40, 122
- [93] G. Grefenstette. Enterprise search trends and challenges. In *CHORUS Final Conf.*, 2009. 1
- [94] P. J. Guo and M. Seltzer. BURRITO: Wrapping Your Lab Notebook in Computational Infrastructure. In *Proc. 4th Usenix Conf. on Theory and Practice of Provenance (TAPP)*, pages 1–4, 2012. 36
- [95] K. A. Gyllstrom, C. Soules, and A. Veitch. Confluence: enhancing contextual desktop search. In *Proc. SIGIR*, page 717, 2007. 17, 66

- [96] G. J. Hahm, J. H. Lee, and H. W. Suh. Semantic relation based personalized ranking approach for engineering document retrieval. *Advanced Engineering Informatics*, 29(3):366—379, feb 2015. 2, 16
- [97] G. J. Hahm, M. Y. Yi, J. H. Lee, and H. W. Suh. A personalized query expansion approach for engineering document retrieval. *Advanced Engineering Informatics*, 28(4):344–359, oct 2014. 2, 16, 39
- [98] E. Hatcher and O. Gospodnetić. *Lucene in Action*. Manning Publications, 2004. 64
- [99] T. Hauer, D. Rogulin, S. Zillner, A. Branson, J. Shamdasani, A. Tsymbal, M. Huber, T. Solomonides, and R. McClatchey. An Architecture for Semantic Navigation and Reasoning with Patient Data - Experiences of the Health-e-Child Project. In *Proc. 7th Int. Conf. on The Semantic Web*, volume 5318, pages 737–750, oct 2008. 123
- [100] D. Hawking. Challenges in enterprise search. In *Proc. 15th Australasian Database Conf.*, volume 27, pages 15–24, 2004. 1, 39
- [101] D. R. Heidi Lam, D. Tang, and T. Munzner. Session Viewer: Visual Exploratory Analysis of Web Session Logs. In *IEEE Symp. on Visual Analytics Science and Technology*, pages 147–154, 2007. 36, 122
- [102] J. Heinrich and D. Weiskopf. State of the Art of Parallel Coordinates. In *Eurographics 2013 - State of the Art Reports*, 2013. 28
- [103] A. Heitzmann, B. Palazzi, C. Papamanthou, and R. Tamassia. Effective Visualization of File System Access-Control. In *Proc. 5th Int. workshop on Visualization for Comp. Security (VizSec)*, pages 18–25, 2008. 26
- [104] M. Hlawatsch, P. Leube, W. Nowak, and D. Weiskopf. Flow radar glyphs—static visualization of unsteady flow with uncertainty. *IEEE Trans. on Visualization & Comp. Graphics*, 17(12):1949–1958, 2011. 31, 97
- [105] O. Hoeber and X. D. Yang. Exploring Web Search Results Using Coordinated Views. In *Int. Conf. on Coordinated and Multiple Views in Exploratory Visualization*, pages 3–13, 2006. 19, 94, 175
- [106] S.-H. Hsieh, H.-T. Lin, N.-W. Chi, K.-W. Chou, and K.-Y. Lin. Enabling the development of base domain ontology through extraction of knowledge from

- engineering domain handbooks. *Advanced Engineering Informatics*, 25(2):288–296, apr 2011. 66
- [107] D. Huynh, D. R. Karger, D. Quan, and V. Sinha. Haystack: A Platform for Creating, Organizing and Visualizing Semistructured Information. In *Proc. 8th Int. Conf. on Intelligent User Interfaces*, page 323, jan 2003. 32, 33, 122, 176
- [108] R. Ichise. Machine Learning Approach for Ontology Mapping Using Multiple Concept Similarity Measures. In *7th IEEE/ACIS Int. Conf. on Comp. and Information Science*, pages 340–346, 2008. 18, 67
- [109] E. Isaacs, K. Domico, S. Ahern, E. Bart, and M. Singhal. Footprints: A Visual Search Tool that Supports Discovery and Coverage Tracking. *IEEE Trans. on Visualization & Comp. Graphics*, 20(12):1793–1802, 2014. 21, 22, 121, 176
- [110] J. Johansson, C. Forsell, M. Lind, and M. Cooper. Perceiving patterns in parallel coordinates: determining thresholds for identification of relationships. *Information Visualization*, 7(2):152–162, 2008. 28
- [111] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *IEEE Conf. on Visualization*, pages 284–291, 1991. 24, 39, 91, 96, 144
- [112] K. S. Jones. a Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, 28(1):11–21, 1972. 145
- [113] A. Kachkaev, J. Wood, and J. Dykes. Glyphs for Exploring Crowd-sourced Subjective Survey Classification. *Comp. Graphics Forum*, 33(3):311–320, 2014. 30
- [114] D. R. Karger and D. Quan. Haystack: a user interface for creating, browsing, and organizing arbitrary semistructured information. In *Proc. CHI*, pages 777–778, 2004. 14, 66
- [115] P. Karnick, S. Jeschke, D. Cline, A. Razdan, E. Wentz, and P. Wonka. A Shape Grammar for Developing Glyph-based Visualizations. *Comp. Graphics Forum*, 28(8):2176–2188, 2009. 97
- [116] A. Karve and M. Gleicher. Glyph-based Overviews of Large Datasets in Structural Bioinformatics. In *11th Int. Conf. on Information Visualization*, pages 1–6, 2007. 30

- [117] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methods—a survey. *ACM Computing Surveys*, 39(4):10–es, 2007. 123
- [118] D. A. Keim. Designing Pixel-Oriented Visualization Techniques: Theory and Applications. *IEEE Trans. on Visualization & Comp. Graphics*, 6(1):59–78, 2000. 26, 91, 96
- [119] A. Kerren, H. C. Purchase, and M. O. Ward. Introduction to Multivariate Network Visualization. In *Multivariate Network Visualization*, pages 1–9. 2013. 26
- [120] C. S. Khoo, B. Luyt, C. Ee, J. Osman, H.-H. Lim, and S. Yong. How users organize electronic files on their workstations in the office environment: a preliminary study of personal information organization behaviour. *Information Research*, 11(2), 2007. 15, 66
- [121] J. Kim and W. B. Croft. Retrieval Experiments Using Pseudo-desktop Collections. In *Proc. CIKM*, pages 1297–1306, 2009. 82, 83
- [122] G. Kindlmann and C.-F. Westin. Diffusion Tensor Visualization with Glyph Packing. *IEEE Trans. on Visualization & Comp. Graphics*, 12(5):1329–1335, 2006. 97
- [123] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49–79, 2004. 17, 66, 146
- [124] M. Kreuseler, T. Nocke, and H. Schumann. A History Mechanism for Visual Data Mining. In *IEEE Symp. on Information Visualization (INFOVIS)*, pages 49–56, 2004. 32, 33, 122
- [125] Laing O’Rourke. Unlocking Digital Engineering. *Engineering Excellence Journal*, page 84, 2014. 142
- [126] B. Laramee, Z. Peng, and R. Laramee. Higher dimensional vector field visualization: A survey. In *Proc. of Theory and Practice of Comp. Graphics*, pages 149–163, 2009. 31

-
- [127] J. Lee, M. Podlaseck, E. Schonberg, and R. Hoch. Visualization and Analysis of Clickstream Data of Online Stores for Understanding Web Merchandising. *Data Mining and Knowledge Discovery*, 5(1-2):59–84, jan 2001. 32, 36, 40, 122
- [128] P. A. Legg, D. H. S. Chung, M. L. Parry, M. W. Jones, R. Long, I. W. Griffiths, and M. Chen. MatchPad: Interactive Glyph-Based Visualization for Real-Time Sports Performance Analysis. *Comp. Graphics Forum*, 31(3):1255–1264, jun 2012. 31, 97
- [129] M. A. A. Leite and I. L. M. R. Ricarte. Fuzzy information retrieval model based on multiple related ontologies. In *20th IEEE Int. Conf. on Tools with Artificial Intelligence*, pages 309–316, 2008. 17, 67
- [130] H.-T. Lin, N.-W. Chi, and S.-H. Hsieh. A concept-based information retrieval approach for engineering domain-specific technical documents. *Advanced Engineering Informatics*, 26(2):349–360, apr 2012. 16
- [131] H. Lü and J. Fogarty. Cascaded Treemaps: Examining the Visibility and Stability of Structure in Treemaps. In *Proc. Graphics Interface*, pages 259–266, 2008. 25
- [132] X. Ma, E. J. M. Carranza, C. Wu, and F. D. van der Meer. Ontology-aided annotation, visualization, and generalization of geological time-scale information from online geological map services. *Computers & Geosciences*, 40:107–119, 2012. 123
- [133] P. Macko and S. Margo. Provenance map orbiter: Interactive exploration of large provenance graphs. In *TaPP*, pages 1–6, 2011. 34, 122
- [134] E. Maguire, P. Rocca-Serra, S.-A. Sansone, and J. Davies. Visual Compression of Workflow Visualizations with Automated Detection of Macro Motifs. *IEEE Trans. on Visualization & Comp. Graphics*, 19(12):2576–2585, 2013. 31, 35, 122
- [135] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies, and M. Chen. Taxonomy-based Glyph Design – with a Case Study on Visualizing Workflows of Biological Experiments. *IEEE Trans. on Visualization & Comp. Graphics*, 18(12):2603–2612, 2012. 29, 31, 35, 97, 122, 177

- [136] C. D. Manning, P. Raghavan, and H. Schutze. *An Introduction to Information Retrieval*. Number c. Cambridge University Press, 2009. 64, 78, 151, 152, 153, 161
- [137] N. Martínez-Bazan, M. Á. Águila-Lorente, V. Muntés-Mulero, D. Dominguez-Sal, S. Gómez-Villamor, and J.-L. Larriba-Pey. Efficient graph management based on bitmap indices. In *Proc. 16th Int. Database Engineering & Applications Sysmp.*, pages 110–119, 2012. 71
- [138] N. Martínez-Bazan, V. Muntés-Mulero, S. Gómez-Villamor, J. Nin, M.-A. Sánchez-Martínez, and J.-L. Larriba-Pey. Dex: high-performance exploration on large graphs for information retrieval. In *Proc. CIKM*, pages 573–582, 2007. 71
- [139] H. Maryam, R. E.-B. Samhaa, and R. Ahmed. A Survey of Ontology Learning Approaches. *Int. Journal of Comp. Applications*, 22(9):36–43, 2011. 18, 67
- [140] G. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review*, 63(2):81–97, 1956. 93
- [141] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995. 71
- [142] T. Montecchi, D. Russo, and Y. Liu. Searching in Cooperative Patent Classification: Comparison between keyword and concept-based search. *Advanced Engineering Informatics*, 27(3):335–345, aug 2013. 16
- [143] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche. The Open Provenance Model core specification (v1.1). *Future Generation Comp. Systems*, 27(6):743–756, 2011. 35
- [144] R. Moskovitch and Y. Shahar. Vaidurya: a multiple-ontology, concept-based, context-sensitive clinical-guideline search engine. *Journal of Biomedical Informatics*, 42(1):11–21, feb 2009. 17, 67
- [145] H. Müller, R. Reihs, S. Sauer, K. Zatloukal, M. Streit, A. Lex, B. Schlegl, and D. Schmalstieg. Connecting Genes with Diseases. In *13th Int. Conf. Information Visualisation*, pages 323–330, 2009. 31, 97

-
- [146] H. Müller, S. Sauer, K. Zatloukal, and T. Bauernhofer. Interactive Patient Records. In *14th Int. Conf. Information Visualisation*, pages 252–257, 2010. 97
- [147] K.-K. Muniswamy-Reddy, U. Braun, D. A. Holland, P. Macko, D. Maclean, D. Margo, M. Seltzer, and R. Smogor. Layering in Provenance Systems. In *Proc. USENIX Annual Technical Conf.*, 2009. 34
- [148] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer. Provenance-Aware Storage Systems. In *Proc. USENIX Annual Technical Conf.*, pages 43–56, 2006. 34
- [149] T. Munzner. *Visualization Analysis and Design*. A K Peters Visualization Series, CRC Press, 2014. 28, 176
- [150] P. Mylonas, D. Vallet, P. Castells, M. Fernandez, and Y. Avrithis. Personalized information retrieval based on context and ontological knowledge. *The Knowledge Engineering Review*, 23(01):73–100, 2008. 66, 84
- [151] R. Navigli and P. Velardi. An analysis of ontology-based query expansion strategies. In *Proc. 14th European Conf. on Machine Learning, Workshop on Adaptive Text Extraction and Mining*, pages 42–49, 2003. 18, 39, 67
- [152] P. Nelson. Search Accuracy Analytics. Technical report, Search Technologies, 2014. 84
- [153] A. Nocaj and U. Brandes. Organizing Search Results with a Reference Map. *IEEE Trans. on Visualization & Comp. Graphics*, 18(12):2546–2555, 2012. 21, 22, 96, 121, 176
- [154] D. Oelke, H. Janetzko, S. Simon, K. Neuhaus, and D. A. Keim. Visual Boosting in Pixel-based Visualizations. *Comp. Graphics Forum*, 30(3):871–880, 2011. 26, 28, 96, 176
- [155] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *Proc. SIGIR*, pages 143–150, 2003. 15, 66
- [156] K. Onak and A. Sidiropoulos. Circular partitions with applications to visualization and embeddings. In *Proc. 24th Annual Symp. on Computational Geometry (SCG '08)*, page 28, 2008. 26, 27, 176

- [157] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, 1998. 1, 39
- [158] G. Pallis, A. Katsifodimos, and M. D. Dikaiakos. Effective Keyword Search for Software Resources Installed in Large-Scale Grid Infrastructures. In *IEEE/WIC/ACM Int. Joint Conf. on Web Intelligence and Intelligent Agent Technologies*, pages 482–489, 2009. 14, 66
- [159] G. Pallis, A. Katsifodimos, and M. D. Dikaiakos. Searching for Software on the EGEE Infrastructure. *Journal of Grid Computing*, 8(2):281–304, 2010. 14, 66
- [160] H. Park, R. Ikeda, and J. Widom. RAMP: A System for Capturing and Tracing Provenance in MapReduce Workflows. In *Proc. 37th Int. Conf. on Very Large Data Bases (VLDB)*, 2011. 33
- [161] Z. Peng, E. Grundy, R. S. Laramée, G. Chen, and N. Croft. Mesh-driven vector field clustering and visualization: An image-based approach. *IEEE Trans. on Visualization & Comp. Graphics*, 18(2):283–298, 2012. 31
- [162] L. R. Peterson and M. J. Peterson. Short-term retention of individual verbal items. *Journal of Experimental Psychology*, 58(3):193–198, 1959. 93
- [163] J. E. Pitkow and K. A. Bharat. WEBVIZ: A Tool for World Wide Web Access Log Analysis. In *Proc. WWW*, pages 271–277, 1994. 33, 122
- [164] J. Pratt, P. V. Radulescu, R. M. Guo, and R. a. Abrams. It’s Alive!: Animate Motion Captures Visual Attention. *Psychological Science*, 21(11):1724–1730, 2010. 93
- [165] A. Pritzkau and D. Bartz. Scattering and Jittering : Using Real and Illusionary Motion for Better Visual Scatterplot Analysis. In *IEEE InfoVis Conf.*, 2010. 97
- [166] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE Trans. on Visualization & Comp. Graphics*, 22(1), 2016. 121
- [167] A. Rajaraman and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, dec 2011. 127

- [168] R. A. Rensink. Failure to see more than one change at a time. *Journal of Vision*, 2(7):245, 2002. 92
- [169] R. A. Rensink, J. K. O'Regan, and J. J. Clark. To see or not to see: The need for attention to perceive changes in scenes. *Psychological Science*, 8(5):368–373, 1997. 92
- [170] P. Rhodes, E. Kraemer, and B. Reed. VisION: an interactive visualization ontology. In *Proc. 44th Annual Southeast Regional Conf.*, pages 405–410. ACM, 2006. 123
- [171] Roberto Tamassia. *Handbook of Graph Drawing and Visualization*. CRC Press, 2013. 26
- [172] J. Roberts, N. Boukhelifa, and P. Rodgers. Multiform glyph based web search result visualization. In *Int. Conf. on Information Visualisation*, pages 549–554, 2002. 19, 30, 94
- [173] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of Animation in Trend Visualization. *IEEE Trans. on Visualization & Comp. Graphics*, 14(6):1325–1332, 2008. 93, 97
- [174] S. Robertson. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2010. 15, 66
- [175] J. J. Rocchio. Relevance feedback in information retrieval. Technical report, 1965. 77, 146
- [176] R. M. Rohrer, J. L. Sibert, and D. S. Ebert. The Shape of Shakespeare: Visualizing Text using Implicit Surfaces. In *IEEE Symp. on Information Visualization*, pages 121–129, 1998. 30, 97
- [177] T. Ropinski, S. Oeltze, and B. Preim. Survey of glyph-based visualization techniques for spatial multivariate medical data. *Computers & Graphics*, 35(2):392–401, 2011. 31, 97
- [178] T. Ropinski and B. Preim. Taxonomy and usage guidelines for glyph-based medical visualization. In *Proc. 19th Conf. on Simulation and Visualization*, pages 121–138, 2008. 31

- [179] T. Ropinski, M. Specht, K. H. H. Jennis Meyer-Spradow, and B. Preim. Surface Glyphs for Visualizing Multimodal Volume Data. In *Proc. 12th Int. Fall Workshop on Vision, Modeling, and Visualization*, pages 3–12, 2007. 31
- [180] R. Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39(19):3157–3163, 1999. 93
- [181] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(2):95–145, jun 2003. 77, 146
- [182] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. 17, 145
- [183] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. J. Moorhead. Noodles: a tool for visualization of numerical weather model ensemble uncertainty. *IEEE Trans. Visualization & Comp. Graphics*, 16(6):1421–30, 2010. 32
- [184] C. Scheidegger, D. Koop, E. Santos, H. Vo, S. Callahan, J. Freire, and C. Silva. Tackling the Provenance Challenge one layer at a time. *Concurrency and Computation: Practice and Experience*, 20(5):473–483, apr 2008. 35, 122
- [185] H.-J. Schulz, S. Hadlak, and H. Schumann. The Design Space of Implicit Hierarchy Visualization: A Survey. *IEEE Trans. Visualization & Comp. Graphics*, 17(4):393–411, 2011. 24, 25
- [186] S. Shah, C. A. N. Soules, G. R. Ganger, and B. D. Noble. Using provenance to aid in personal file search. In *USENIX Annual Technical Conf.*, pages 1–14, 2007. 15, 66
- [187] Z. Shen, J. Wei, N. Sundaresan, and K.-L. Ma. Visual analysis of massive web session data. In *IEEE Symp. on Large Data Analysis and Visualization*, pages 65–72, oct 2012. 36, 122
- [188] C. Shi, Y. Wu, S. Liu, H. Zhou, and H. Qu. LoyalTracker: Visualizing Loyalty Dynamics in Search Engines. *IEEE Trans. on Visualization & Comp. Graphics*, PP(99):10, 2014. 36, 122
- [189] L. Shi and R. Setchi. Ontology-based personalised retrieval in support of reminiscence. *Knowledge-Based Systems*, 45:47–61, 2013. 17, 67, 82

-
- [190] B. Shneiderman. Tree Visualization with Tree-Maps - 2-D Space-Filling Approach. *ACM Trans. on Graphics*, 11(1):92–99, 1992. 24, 39
- [191] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *IEEE Symp. on Visual Languages*, page 336, 1996. 99, 132
- [192] B. Shneiderman and C. Plaisant. Treemaps for space-constrained visualization of hierarchies. www.cs.umd.edu/hcil/treemap-history. 25, 26
- [193] C. T. Silva, J. Freire, and S. P. Callahan. Provenance for Visualizations: Reproducibility and Beyond. *Computing in Science & Engineering*, 9(5):82–89, sep 2007. 35
- [194] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. Analysis of a Very Large AltaVista Query Log. In *Proc. SIGIR*, pages 6–12, 1999. 19
- [195] D. J. Simons and R. A. Rensink. Change blindness: Past, present, and future. *Trends in Cognitive Sciences*, 9(1):16–20, 2005. 92
- [196] C. A. N. Soules and G. R. Ganger. Connections: using context to enhance file search. In *Proc. 20th ACM Symp. on Operating Systems Principles*, volume 39, page 119, 2005. 15
- [197] A. Soylu, M. G. Skjæveland, M. Giese, I. Horrocks, E. Jimenez-Ruiz, E. Kharlamov, and D. Zheleznyakov. A Preliminary Approach on Ontology-Based Visual Query Formulation for Big Data. In *Metadata and Semantics Research*, volume 390, pages 201–212. Springer, 2013. 123
- [198] B. Stefan and C. L. a. Clarke. Indexing Time vs . Query Time Trade-offs in Dynamic Information Retrieval Systems. *Proc. CIKM*, pages 317–318, 2005. 161
- [199] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: A Study of Orienteering Behavior in Directed Search. In *Proc. CHI*, pages 415–422, 2004. 15, 66
- [200] R. Thiagarajan, G. Manjunath, and M. Stumptner. Computing semantic similarity using ontologies. In *Int. Semantic Web Conf.*, pages 1–16, 2008. 127

- [201] P. Thomas. Server characterisation and selection for personal metasearch. In *Proc. SIGIR*, volume 42, pages 106–108, 2008. 15, 66
- [202] Y. Tu and H.-W. Shen. GraphCharter: Combining browsing with query to explore large semantic graphs. In *IEEE Pacific Visualization Symp.*, pages 49–56, 2013. 123
- [203] D. Turnbull and J. Berryman. *Relevant Search*. Manning Publications, 2015. 64
- [204] D. Turo and B. Johnson. Improving the visualization of hierarchies with treemaps: design issues and experimentation. In *IEEE Conf. on Visualization*, pages 124–131, 1992. 24
- [205] B. Tversky, J. B. Morrison, M. Betrancourt, and J. Hall. Animation: can it facilitate? *Int. Journal of Human-Computer Studies*, 57(4):247–262, 2002. 97
- [206] UK Cabinet Office. Government Construction Strategy, 2011. 2, 4, 121
- [207] D. Vallet, M. Fernandez, and P. Castells. An Ontology-Based Information Retrieval Model. In *The Semantic Web: Research and Applications*, volume 3532 of *LNCS*, pages 455–470. Springer, 2005. 17, 66
- [208] J. J. Van Wijk, H. Van de Wetering, and H. de Wetering. Cushion treemaps: visualization of hierarchical information. In *IEEE Symp. on Information Visualization*, pages 73–78, 1999. 25, 39
- [209] R. Walker, A. Slingsby, J. Dykes, K. Xu, J. Wood, P. H. Nguyen, D. Stephens, B. L. W. Wong, and Y. Zheng. An extensible framework for provenance in human terrain visual analytics. *IEEE Trans. on Visualization & Comp. Graphics*, 19(12):2139–48, dec 2013. 35, 122
- [210] S. Walton, K. Berger, J. Thiyagalingam, B. Duffy, H. Fang, C. Holloway, A. E. Trefethen, and M. Chen. Visualizing Cardiovascular Magnetic Resonance (CMR) imagery: Challenges and opportunities. *Progress in Biophysics and Molecular Biology*, 115(2-3):349–358, aug 2014. 31
- [211] W. Wang, C. Peery, A. Marian, T. D. Nguyen, and I. C. Society. Efficient Multidimensional Fuzzy Search for Personal Information Management Systems. *IEEE Trans. on Knowledge and Data Engineering*, 24(9):1584–1597, 2012. 15, 66, 82, 85

- [212] M. O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3-4):194–210, dec 2002. 29, 30, 97
- [213] M. O. Ward. Multivariate data glyphs: Principles and practice. In *Handbook of Data Visualization*, pages 179–198. Springer Berlin Heidelberg, 2008. 97
- [214] C. Ware and R. Bobrow. Motion coding for pattern detection. In *Proc. 3rd Symp. on Applied Perception in Graphics and Visualization*, pages 107–110, 2006. 97
- [215] C. Ware and M. D. Plumlee. Designing a better weather display. *Information Visualization*, 12(3-4):221–239, dec 2012. 32
- [216] S. Waterson, J. I. Hong, T. Sohn, J. Heer, T. Matthews, and J. Landay. What Did They Do? Understanding Clickstreams with the WebQuilt Visualization System. In *Proc. Working Conf. on Advanced Visual Interfaces*, pages 94–102, 2002. 33, 122
- [217] A. Watson. Digital buildings - Challenges and opportunities. *Advanced Engineering Informatics*, 25(4):573–581, 2011. 2
- [218] M. Wattenberg. A note on space-filling visualizations and space-filling curves. In *IEEE Symp. on Information Visualization (InfoVIS'05)*, pages 181–186, 2005. 26, 27, 176
- [219] J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma. Visual cluster exploration of web clickstream data. In *IEEE VAST*, pages 3–12, oct 2012. 32, 36, 40, 122
- [220] K. Wetzel. Pebbles—Using Circular Treemaps to Visualize Disk Usage, 2003. 26
- [221] C. M. Wittenbrink, A. T. Pang, and S. K. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Trans. on Visualization & Comp. Graphics*, 2(3):266–279, 1996. 31, 97
- [222] M. Wohlfart and H. Hauser. Story telling for presentation in volume visualization. In *Proc. 9th Joint Eurographics / IEEE VGTC Conf. on Visualization*, pages 91–98, 2007. 97

- [223] K. Wongsuphasawat and D. Gotz. Exploring Flow, Factors, and Outcomes of Temporal Event Sequences with the Outflow Visualization. *IEEE Trans. on Visualization & Comp. Graphics*, 18(12):2659–2668, dec 2012. 34
- [224] J. Wood and J. Dykes. Spatially Ordered Treemaps. *IEEE Trans. on Visualization & Comp. Graphics*, 14(6):1348–1355, nov 2008. 96, 148
- [225] H. Xiao and I. F. Cruz. A Multi-Ontology Approach for Personal Information Management. In *Semantic Desktop Workshop*, 2005. 18, 67
- [226] Z. Xu, M. Karlsson, C. Tang, and C. Karamanolis. Towards a semantic-aware file store. In *Proc. 9th Conf. on Hot Topics in Operating Systems*, page 31, 2003. 15, 66
- [227] J. Yang, D. Hubball, M. O. Ward, E. A. Rundensteiner, and W. Ribarsky. Value and relation display: interactive visual exploration of large data sets with hundreds of dimensions. *IEEE Trans. on Visualization & Comp. Graphics*, 13(3):494–507, 2007. 97
- [228] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated Exploration of Dynamic Graphs with Radial Layout. In *IEEE Symp. on Information Visualization*, page 43, 2001. 105
- [229] J. Zhang, C. Chen, M. S. Vogeley, D. Pan, A. Thakar, and J. Raddick. SDSS Log Viewer: visual exploratory analysis of large-volume SQL log data. In *Proc. Visualization and Data Analysis*, volume 8294, page 13, 2012. 32, 36, 40, 122
- [230] X. Zhang, X. Hou, X. Chen, and T. Zhuang. Ontology-based semantic retrieval for engineering domain knowledge. *Neurocomputing*, 116:382–391, sep 2013. 18, 67
- [231] Y. Zhang, P. J. Passmore, and R. H. Bayford. Visualization of multidimensional and multimodal tomographic medical imaging data, a case study. *Philosophical Trans. of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367(1900):3121–3148, aug 2009. 97
- [232] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *Proc. WWW*, page 1039, 2006. 126, 145

- [233] J. Zhao, P. Forer, and A. S. Harvey. Activities, ringmaps and geovisualization of large human movement fields. *Information Visualization*, 7:198—209, 2008. 97
- [234] L. Zhao and J. Callan. A generative retrieval model for structured documents. In *Proc. CIKM*, pages 1163–1172, 2008. 15, 66
- [235] L. Zhou. Ontology learning: state of the art and open issues. *Information Technology and Management*, 8(3):241–252, 2007. 18, 39, 67