Vrije Universiteit Brussel

FACULTY OF SCIENCE AND BIO-ENGINEERING SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Enhanced Retrieval and Discovery of Desktop Documents

Graduation thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Applied Sciences and Engineering: Computer Science

Trung Ngoc Tran

Promoter: Prof. Dr. Beat Signer
Advisor: Ahmed A. O. Tayeh

Academic year 2014-2015

Abstract

With recent advances in technology, it is becoming more convenient and affordable to accommodate thousands of documents. The rapid increase in the overall volume of information has exposed significant challenges regarding retrieval tasks, especially on the computer desktop, since modern desktop searches insufficiently support the retrieval of items from documents. Indeed, documents do not exist in isolation, but rather pertain to each other. The relationships between documents can be implicit due to similarities between the document content and metadata, or explicit due to the links (associations) created by a cross-document link service. Through a review of different techniques for document retrieval on the desktop, we have identified fundamental flaws of the built-in tools, including: (i) the static hierarchical structure that imitates the physical desktop, (ii) the lack of automatic components in document management and (iii) the insufficient search features that limit advanced search compositions (e.g. boolean queries and wildcards). We also examined different attempts to enhance document retrieval and discovery on the desktop and classified them into three categories according to approach: (i) those that provide different visualisation methods of the hierarchy, (ii) those that change to a non-hierarchical structure and (iii) those that move towards the memex using associative trails.

Based on the findings of our review, we have developed a novel model that supports the retrieval of documents and overcomes many of the shortcomings of document retrieval systems on the desktop. In our model, the document content and metadata are used to establish implicit relationships between documents using data mining algorithms. The explicit links are considered to enhance the retrieval of documents. Furthermore, users can establish advanced search queries by means of boolean queries and wildcards. We have proposed an extensible architecture with two main advantages. First, the system is not limited to a specific document format, but is compatible with both existing and emerging document formats via a plug-in mechanism. Second, the visualisation component of the system can be extended to support any kind of visualisation (e.g. the schemaball and the ordinary list). We have also developed a prototype as a proof of concept for the proposed solution. The prototype has been evaluated by end users to obtain feedback about the proposed solution.

Declaration of Originality

I hereby declare that this thesis was entirely my own work and that any additional sources of information have been duly cited. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this thesis has not been submitted for a higher degree to any other University or Institution.

Acknowledgements

I would like to express my immeasurable appreciation and deepest gratitude to those who supported me in different ways during the course of this thesis.

First and foremost, I own deep gratitude to my supervisor Ahmed Tayeh, for his valuable comments, suggestions and advices that contributed to this thesis. I have worked directly with him since September 2014 and since then, I have learned from him that the research is possible with perseverance and focus. Without his persistent support, this thesis would have not been accomplished.

I am very much thankful to my promoter Prof. Dr. Beat Signer who, while burdened with countless responsibilities, found time to proofread parts of my thesis, and give valuable comments.

Last but not least, it is my privilege to thank Belgian Development Agency for granting me the scholarship to pursuit the Master study at Vrije Universiteit Brussel. I treasure precious moments and profound knowledge this opportunity has given me.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

"Data smog" is the term coined by David Shenk in his book of the same title that presents the big picture of information evolution in the last few years [74]. According to the book, with the advance of information technology, individuals would face the problem of the overwhelming amount of data, resulting in difficulties in finding and retrieving information. In an attempt to assist the user in managing documents, existing computer systems, including Microsoft Windows and Apple Macintosh, organise them in nested folders. The user either navigate in a tree-like structure or insert keywords in the search form to find documents.

Back in 1965, Ted Nelson introduced the concept of the *zippered list* to give structure to personal documents [55]. Nelson's project was inspired by the idea of Vannevar Bush in the paper As We May Think [6], in which Bush proposed to use associative trails between documents. So far there are several researches such as Xanadu [55], oN-Line System [21] and NoteCards [28] and the resource- selector-link metamodel (RSL) [75] working towards Bush's vision. The goals of these researches are to support hyperlinks between different types of media and reinforce document semantics through annotations.

To us, it is intriguing when the user is enabled to make trails between their

documents because it allows the user to "jump" from document to document in any flexible way could imagine. The associative trails between documents, if they can be exploited, can help us enhance the retrieval task on the desktop.

1.2 Problem Statement

Typically there are two methods for the user to search for documents on the desktop. The user either traverses the hierarchical tree to find the documents or uses the built-in search (e.g., Windows Search, Mac OS Spotlight) to retrieve the document. In the former method, most of the time the system forces the user to recall their memory when they go through folders hierarchically to find the file [23]. The user also have problems with categorising and retrieving documents later because the static directories are inadequate to document organisation. Instead, documents should be grouped dynamically according to their contents and the needs of the user. The latter method is more flexible because the user can find documents directly. Most of the time, the user will search for documents by *title matching* unless they tailor their search by *content matching*. However, the content matching search mechanism is still limited in the sense that it only returns documents that contain the exact input keywords. However, the words usually have multiple forms derived from a single root form. For example, the string "*computing*" or "*computer*" can be derived from the root form "*compute*". Thus, the system searches can neglect possible documents that are relevant to the user queries when using simple keyword matching. Another issue with the system searches is that they do not take into account the similarity between documents to enhance document retrieval and discovery on the desktop.

Due to the these issues, there are efforts that try to enhance retrieval and discovery task of the user on the desktop. One good example of research that takes into account document similarity to improve document navigation is DeFiBro [54]. However, DeFiBro has a critical issue because it solely focuses on exploiting the document properties (for example, author and number of words) and omits the content of the documents.

More importantly, there is a recent research called the cross-document link service that offers a novel architecture allowing the user to establish explicit links between different documents from different document formats [84]. While the idea of creating explicit links between documents is intriguing, it is so new that no effort is being made towards exploiting the explicit links to enhance document retrieval and discovery task.

Whereas the novel idea of creating implicit links between documents using document content as well as taking the use of explicit links promises

better support for retrieving and discovering documents, it exposes critical questions: *how should we visualise both explicit and implicit links in a single interface?*, and *what framework supports such visualisation?* Indeed, both issues stem from the fact that we lack a framework that not only supports visualising both kinds of links but also needs to be open enough to embrace multiple types of visualisation.

1.3 Research Objectives

The goal of this thesis were as follows.

- *Review the state-of-the-art methods* in desktop retrieval that support the user. The methods are either system-provided tools or additional models using machine learning techniques.
- Investigate how we can use implicit links between documents using text mining algorithms and explicit links to support an innovative way for document retrieval on the desktop.
- *Investigate an extensible architecture* to support the goal. The architecture is designed to be extensible in terms of supporting multiple visualisations as well as multiple text mining algorithms.
- *Implement a prototype as a proof of concept of the proposed model.* The implementation demonstrates how document retrieval is enhanced using automatic document categorisation as well as explicit links created by the link service. The implementation also helps us reveal benefits and challenges of advanced search features (i.e., complex Boolean and wildcard queries).
- *Conduct an evaluation with end-users* to gather feedback and measure the user experience of the proposed solution.

1.4 Thesis Structure

The structure of this thesis is as follows.

- **Chapter 2: Background**
  This chapter lays the foundation of knowledge for what we will discuss in the following chapters.

- **Chapter 3: Literature Review**
  In this chapter, we review document access activities on the desktop using hierarchical structures. Next, a comprehensive review of different visualisation techniques applied to the existing file hierarchy is represented. Finally, we discuss novel studies focusing on replacing the desktop metaphor with nonhierarchical file structures.

- **Chapter 4: Enhanced Document Retrieval**
  This chapter describes the requirements we have defined for our proposed model and the extensible architecture we formulate to satisfy these requirements.

- **Chapter 5: Implementation**
  In this chapter, we discuss the implementation of our prototype as a proof of concept for the proposed model.

- **Chapter 6: Evaluation**
  This chapter explains the evaluation that we conducted with end-users.

- **Chapter 7: Discussion and Future Work**
  We finalise our works with a discussion and highlight possible future work.

2 Background

This section is to provide the essential background on information retrieval (IR) and clustering techniques. We begin with an overview on IR to see key developments in the past years. We then examine a modelling technique for the document in the digital space, called the Vector Space Model (VSM). The knowledge of the VSM is fundamental to understand how it is possible to measure the similarity between document vectors. We will briefly review some distance measures and then move to the scoring and term weighting. The scoring and term weighting is essential in IR, where every term in the document will be assigned a weight reflecting how important the term is to the whole document. The two last sections are about clustering, where we review how clustering can contribute to IR.

2.1 Information Retrieval

The activities of IR are broad. In the physical space, information retrieval is concerned with normal activities like finding the correct journals in university libraries. When moving from the physical space to the digital space, it is more common to use built-in search tools on personal computers or online search

engines, such as Google[1], Bing[2], Yahoo![3] when it comes to the World Wide Web.

According to [9], IR is an activity of "*finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)*".

2.1.1 A Brief Look at IR

The Dewey decimal classification [71] is a classical example of how documents can be indexed in the physical space. In 1950s, Taube developed coordinate indexing, which uses "uniterm" to organise document collection [83]. The idea is to index documents using keywords. The use of keywords seems to be trivial today but it is considered as a fundamental methodology of document indexing until now. Later, the researchers started to use the probabilistic methods in IR. Some examples are the boolean retrieval model, where queries are boolean expressions [9]; or document scoring, where each document is assigned a specific score regarding to a given query [47]. The concept of *term frequency* was coined around the same time when [51] took into account the usefulness of the frequency of a word occurrence when indexing a document. One of the prominent systems built during this decade is the mechanised IR system for the technical library of General Electric's Aircraft Gas Turbine Division in 1953 [1]. The system uses keyword for the coordinate indexing system where the keywords were manually extracted from parts of documents by professional librarians.

Next, a different methodology was proposed by [82] in 1960s where both documents and queries are modelled by the vectors. The vector model became popular and later was adopted by [69] to define the VSM, where the *cosine similarity* metric was used to measure the angle between vectors. In addition to the VSM, Robertson proposed the principle of probability ranking as an alternative model [65]. The principle depends on the assumption that the relevance of a document to a query is not influenced by other documents in the same collection. Therefore, the rank of a document can be set according to a specific query using the defined criterion variables.

Another significant concept in IR was coined in the 1970s when [77] defined *inverse document frequency* (idf) based on the idea of term frequency. Following the concept of inverse document frequency is the definition of *term frequency - inverse document frequency* (tf-idf) [70]. Later, the concept of

[1]https://www.google.com
[2]http://www.bing.com
[3]https://search.yahoo.com

latent semantic indexing (LSI) was introduced, where the implicit higher order in the semantic structure of documents was taken into account [16]. LSI gave encouraging results when dealing with synonymy problems, but not really well with polysemy problems.

In the 1990s, the World Wide Web was born and thrived vigorously. Along with the development of the Web was the compelling for a Web search engine in the middle of the 1990s. The links between the web pages are an interesting characteristic of the Web, by which the Web page creators can create references from their Web pages to other Web pages. This convention was fully exploited by Sergey Brin and Lawrence Page when they built a prototype of Google [4], and later when they developed the PageRank algorithm to measure the score of Web pages [44].

2.2 Text Indexing

The inverted index is heavily used in both physical and digital text search methods. In a nutshell, an inverted index is a sorted hashmap that allows fast lookup from a word to a document or a document page. In digital IR, the inverted index is built after a number of steps.

1. Provide an identifier for each document
2. For each document, the text will be tokenised, turning the document to a bag of tokens.
3. The next step is optional, where we can process tokens using stemming algorithms and remove stop words and so on.
4. We index documents in the collection using the inverted index.

In Figure 2.1, first, two documents are tokenised and organised into pairs of terms and document IDs. Similar to the inverted index in physical textbooks, it is necessary to sort the inverted index in the digital format, turning the existing inverted index into a sorted list of pairs in the middle column of Figure 2.1. Next, instances of the same term are merged and organised into a dictionary consisting of terms and postings (the right column of Figure 2.1). Together with the corresponding document IDs, some extra information, such as the document frequency of the given term, can be recorded.

Regarding the performance of the inverted index, a number of concerns are given [13].

- **Block update speed** required time to index a document collection.

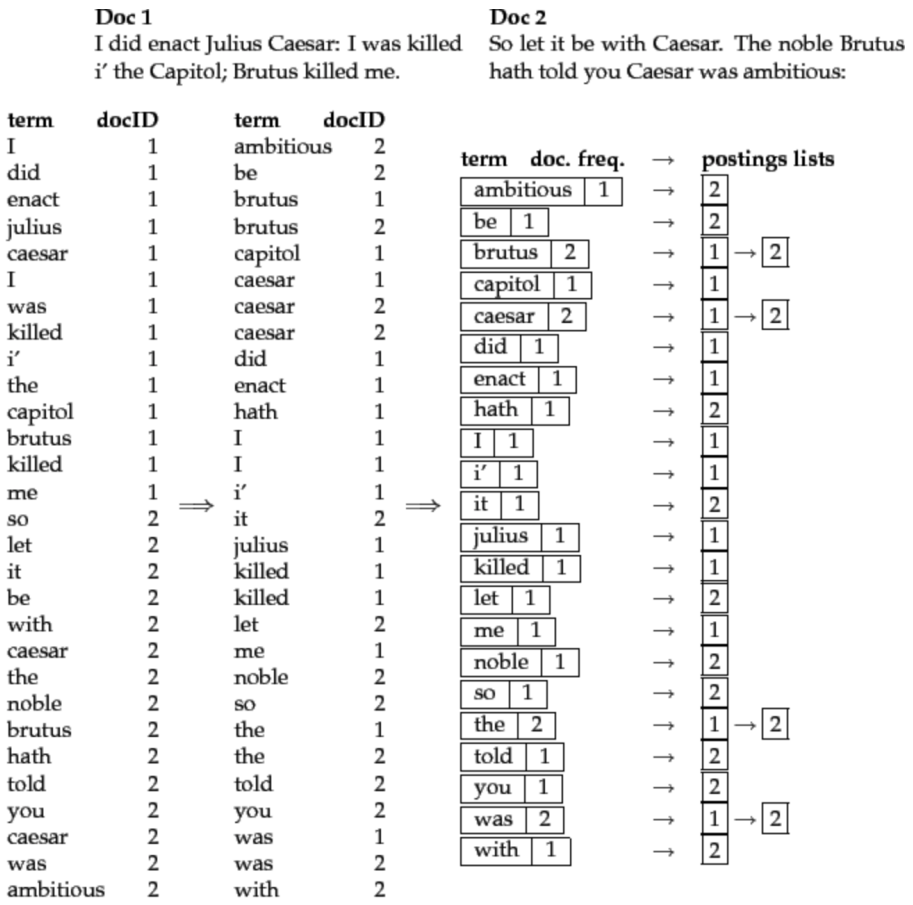


Figure 2.1: Building an index by sorting and grouping [9]

- **Access speed** required time to access the postings for a given word.
- **Index size** the amount of the space taken by the index.
- **Dynamics** how easy the index copes with changes.
- **Scalability** how the inverted index copes with the growing amount of documents.

2.3 Vector Space Model

For a given document D_i, [69] modelled the given document by using a t-dimensional vector.

$$D_i = (d_{i1}, d_{i1}, d_{i1}, ..., d_{il})$$

where d_{ij} represents the weight of the jth term.

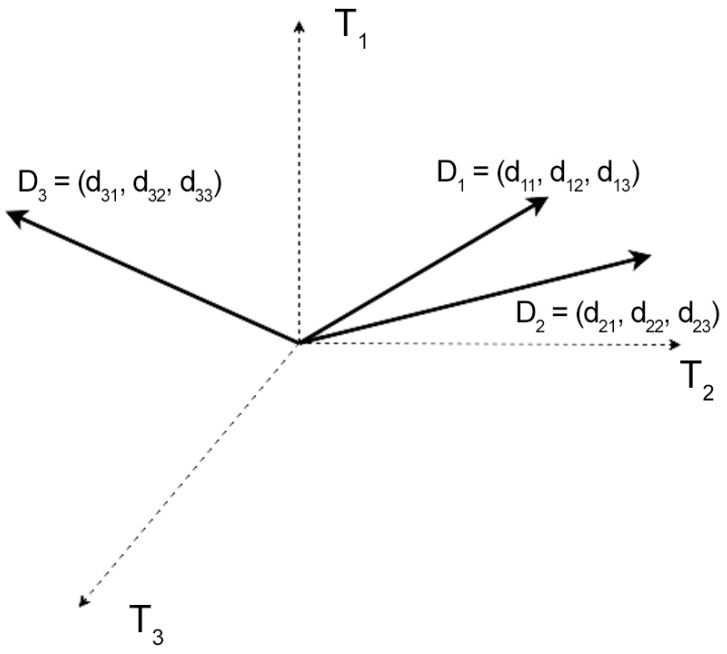


Figure 2.2: An example of using vector space model to represent documents

Consequently, the *corpus* or the collection of documents corresponds to a term-document matrix of size (m x n) where m is the number of unique terms or the *index term* extracted from the corpus and n is the number of documents. It is noted that one potential disadvantage of the vector model is that it contains no information about the word order. Another disadvantage is that the vector model is often called the *sparse vector* since the number of index terms of the corpus usually exceeds the number of terms for a single document. Figure 2.2 is an illustration of the vector representation in 3D space where three documents are modelled using vectors with three components.

2.3.1 Building Term Vocabulary

The process of building term vocabulary for the vector space model requires primitive steps.

- **Filtering** In the document, there are characters and punctuation that do not hold any discriminative power and, therefore have to be removed. The process of eliminating these characters from the corpus is the filtering process.

- **Tokenisation** Tokenisation is the process that takes a stream of strings as an input, and chops down it into smaller parts, called *tokens*. Tokens may include words, phrases or other meaningful parts. Whitespace and punctuation are usually removed through this process.

- **Removing Stopwords** Stopwords are commonly used words in documents. The words such as *a, an, the, was* are examples of stopwords in English. Stopwords often need to be dropped because they do not help to distinguish between documents, but in some exceptional cases, they may help to convey the correct meaning of words, e.g., it would be necessary to have the word *the* when searching for the book *The Lover*. An example of a stemming algorithm is the Porter [56].

- **Stemming** Depending on the grammar context, words may have various forms (e.g., *activity, activities*). These words should have similar basic meaning and origin from basic root form. The goal of stemming process, therefore, is to reduce these inflectional forms [34].

- **Pruning** The process of pruning often uses a threshold value to remove terms that have a low frequency. This process is based on the assumption that the terms whose frequency below the given threshold may form meaningless clusters [49].

- **Adding synonym** One additional step when building term vocabulary is to add synonym by using Wordnet[52]. However, the contribution of using the WordNet ontology toward the clustering process is still ambiguous[33, 72].

2.4 Distance Measure

In Section 2.3, we have discussed the vector space model to understand how documents are represented using the algebraic model. Indeed, the vector space model lays a foundation for further developments in IR. One of them is the distance measure or the similarity measure.

In a nutshell, the distance measure or the similarity measure is to quantify the similarity between document vectors. In this section, we will cover a number of distance measure methods, including Euclidean distance measure,

squared-Euclidean distance measure, Manhattan distance measure and cosine distance measure.

2.4.1 Euclidean Distance Measure

The Euclidean distance d between two points $x = (x_1, x_2, x_3, ..., x_n)$ and $y = (y_1, y_2, y_3, ..., y_n)$ whose x_n and y_n are terms in Euclidean space with n dimensions is given by

$$d(x, y) = d(y, x) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Based on the formula, the Euclidean distance is the length of line segment $\overline{xy}$ between two points x and y.

2.4.2 Squared Euclidean Distance Measure

The squared Euclidean distance measure can be obtain by squaring the standard Euclidean distance. Therefore, we have

$$d^2 = (x - y)^2 = \sum_{i=1}^{n} (x_i - y_i)^2$$

Using Squared Euclidean distance, we can increase weights of objects in the further distance. Also, because the squared Euclidean distance does not take the square root, it can be computed faster than the standard Euclidean distance measure.

2.4.3 Manhattan Distance Measure

The Manhattan distance between two points $x = (x_1, x_2, x_3, ..., x_n)$ and $y = (y_1, y_2, y_3, ..., y_n)$ is the total sum of paths in each dimension. It could be obtained by absolute values of differences between corresponding terms.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

2.4.4 Cosine Distance Measure

Unlike the Euclidean or Manhattan distance measure, the cosine distance measure assumes points as vectors whose components are terms of the points.

Suppose we have two points $x = (x_1, x_2, x_3, ..., x_n)$ and $y = (y_1, y_2, y_3, ..., y_n)$ in the space with n dimensions. From these two points, we can create two vectors $\overrightarrow{a}(x_1, x_2, x_3, ..., x_n)$ and $\overrightarrow{b}(y_1, y_2, y_3, ..., y_n)$ where a_n and b_n are terms of the corresponding points. Then, the dot product of two vectors are given by

$$\overrightarrow{a}.\overrightarrow{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + ... + a_n b_n$$

We can also have

$$\overrightarrow{a}.\overrightarrow{b} = ||\overrightarrow{a}||||\overrightarrow{b}||\cos(\theta)$$

where θ is the angle between vectors $\overrightarrow{a}$ and $\overrightarrow{b}$.

Then we can obtain the cosine of the angle between vectors

$$\cos(\theta) = \frac{\overrightarrow{a}.\overrightarrow{b}}{||\overrightarrow{a}||||\overrightarrow{b}||}$$

Based on the formula, we can see that this metric only shows the difference in the orientation of vectors, not the magnitude of them. Therefore, the cosine distance measure can be used to measure similarity between documents with different lengths in term space.

2.5 Scoring and Term Weighting

A common scenario in any IR model is that the user often create a query and the system will return a list of results accordingly. In order to do that, often the system relies on a *term weighting component* that is in charge of computing scores demonstrating the match between the query and each document.

Indeed, the scoring requires a few steps. Firstly, we can weight each term in the document by many methods. There are several possible weighting schemes we can use. The most naive is to rely on the number of occurrences of the term in the document, or we call *term frequency*. Aside from term frequency, the *inverse document frequency* (tf) and *term frequency and inverse document frequency* (tf-idf) are the additional weighting factors that

contribute to a good weighting scheme [68]. In principle, when we know the weighting score of all terms, we can compute the matching score for the each document in the collection and the query. There are multiple methods that help we decide the matching score (for example, the cosine distance or the Euclidean distance) we will discuss closely in this section.

2.5.1 Term Frequency

As said, the term frequency *tf* is the most naive weighting scheme for terms. The term frequency plainly takes into account the number of occurrences of a term t in a document d. Since the term frequency of a term heavily relies on the length of the document and often each document has different length, it is plausible to divide the term frequency by the document length.

$$tf(t,d) = \frac{f_{t,d}}{l_d}$$

where $f_{t,d}$ is the frequency of term t in document d, l_d is the length of the document d or the total number of occurrences of all terms in document d.

2.5.2 Inverse Document Frequency

When using the term frequency, we implicitly treat all terms equally. Nevertheless, there is a number of terms that may have little power compared to other terms, though they appear more frequently in the corpus. For instance, a collection of books about Java will likely use the term "*java*" exhaustively, but indeed the word "*java*" does not help to distinguish documents.

To weight down the effect of such terms, we define the inverse document frequency idf_t where the *document frequency* df_t indicates the number of documents in which the term t occurs and N the total number of documents [77].

$$idf_t(t,d) = log\frac{N}{df_t}$$

The equation implies the inverse document frequency gives low scores to terms appearing frequently across the corpus and high score to rare terms.

2.5.3 Term Frequency - Inverse Document Frequency

The term frequency and inverse document frequency (tf-idf) is a further step from the idf when the tf-idf takes into consideration the frequency of term as well as the inverse proportion of term in the entire corpus.

$$tf - idf_{t,d} = tf_{t,d} \, x \, idf_t$$

This metric favors the term that has high term frequency in a document and low frequency in the whole collection. We call such term is the *term discriminator* since it helps to distinguish documents from the rest.

2.5.4 Latent Semantic Analysis

Latent Semantic Analysis (LSA) designed to determine correlations in meaning between words and passages. The underlying idea is that words in a large text of corpus have mutual constraints to each others.

LSA relies on Singular Value Decomposition (SVD), which is used to decompose original data into sub-components.

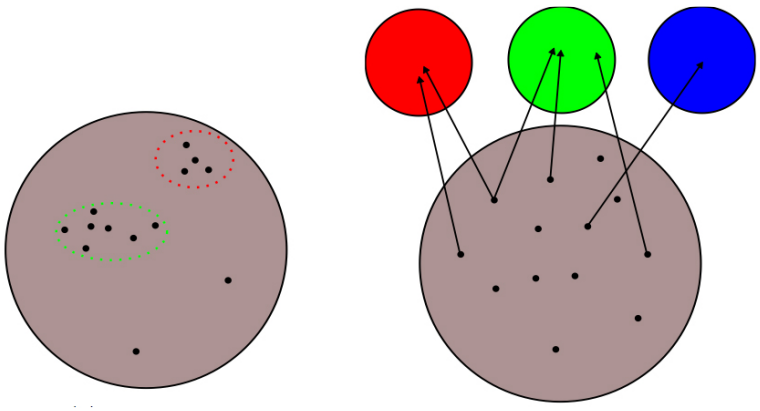
$$M = USV^T$$

in which M is the original matrix, U and V is the orthogonal and normalized, S is a diagonal.

2.6 Clustering

From time to time, the primitive tasks of human beings are organising data into classes and assigning unknown things into existing groups. By doing so, people can understand the meaningful structure of data and relationships between objects. Indeed, the former task is clustering and the latter is classification.

In general, classification is *supervised learning*, whereas clustering is unsupervised learning. Supervised learning means that we have to predefine the classes before assigning unknown objects. Unsupervised learning, in contrast, is to group similar objects together without predefined classes. Figure 2.3 clearly illustrates this distinction.



(a) Clustering (b) Classification

Figure 2.3: Clustering is about grouping unknown objects together whereas classification assigns uncategorized objects into predefined labels [73]

2.6.1 What is a Cluster?

The definition of a cluster has evolved over time. According to a study, a cluster is the representation of mixtures of multivariate normal populations [53]. This definition was adopted by researchers when they designed clustering algorithms. When the mixtures are not constrained, the clustering algorithms can create overlapping clusters. The definition of *ultrametric* based on the concepts of the single linkage and the complete linkage was also used in the generation of some cluster structure [36]. More importantly, most hierarchical clustering routines invoke the ultrametric inequality, and therefore the algorithms can recover the space structure. The definition of a cluster was made clearer when supplying definitions, such as the distance or the similarity, was provided.

According to [76], a typical cluster has five following properties.

- *Separation* describes how much the cluster overlaps the space.
- *Density* describes the compactness of data points.
- *Variance* describes the degree of dispersion of the points within the cluster. The distance is calculated from the centroid of the cluster to the points.
- *Dimension* can be measured as the radious of a cluster but the cluster is not necessarily a circle.

- *Shape* of a cluster could be many types, including ellipsoids or hyperspheres.

The notion of a cluster is variant. Indeed, there are different types of clusters.

- **Well-separated** In this case, the cluster can be decided by the distance from one point to another point in the cluster.
- **Prototype-based** When the data has continuous attributes, the prototype of a cluster is the cluster centroid or the mean of all points in the cluster. For data that has categorical attributes, the alternation is often the medoid of the cluster or the most representative point.
- **Graph-based** In this case, the data often represents an undirect graph where the points represent the nodes of the graph, and the links between the nodes represent the vertices. This definition implies that there would be many vertices within a cluster.
- **Density-based** The compactness of points in the cluster would be higher than the remaining of the data.
- **Shared-property** Objects within a cluster usually share common properties.

2.6.2 Challenges

There are many considerable factors contributing to a good cluster. In loosely speaking, a good cluster can encompass points that are similar to each other in the clusters, and different from points of other clusters. To make a good cluster, there are some challenges. First, the feature selection needs to be considered. When it comes to document clustering, many clustering methods model documents as the vectors. Secondly, clustering methods can be applied in different domains, then picking the most suitable clustering methods is challenging.

2.7 Clustering Algorithm

Clustering methods belong to a number of types.

- **Hierchical clustering** is a type of clustering where clusters can be nested into each other.

- **Partional clustering** is a type of clustering where clusters do not overlap each other.

2.7.1 Hierarchical Clustering

This clustering method will produce clusters in form of the hierarchy, where one cluster can be nested into another. There are disadvantages of the hierarchical clustering over the partitional clustering. Firstly, the hierarchical clustering does not scale because of the merging or splitting phase. Therefore, it is not suitable for real-time applications as well as large collections. More importantly, the hierarchical clustering is slower than the partitional clustering [81]. Typically, there are two types of strategies to produce the hierarchical clustering.

- **Agglomerative** The cluster tree will be produced from bottom to top where child clusters are merged to create parent clusters.
- **Divisive** In contrast to the agglomerative approach, the divisive approach will produce the cluster tree from top to bottom where one parent cluster is divided into child clusters.

In both strategies, there are necessary distance measures and linkage criteria to help deciding the merging or splitting of clusters. Aside from the distance measures we discussed, there are typically some linkage criteria:

- **Single Linkage** In the single linkage clustering, the similarity between two clusters is decided by the similarity of the closest objects or the minimum distance between two objects in two clusters.

$$D(X, Y) = min(d(x, y))$$

  where X and Y are two clusters and x is the object in the cluster X and y is the object in the cluster Y.

- **Complete Linkage** In contrast with the single linkage, the complete linkage is decided by the least similar objects or the farthest distance between objects in two clusters.

$$D(X, Y) = max(d(x, y))$$

  where X and Y are two clusters and x is the object in the cluster X and y is the object in the cluster Y.

- **Average Linkage** The average linkage clustering is the average of distances between pairs of objects in two clusters.

$$D(X, Y) = \frac{1}{|A| \ . \ |B|} \sum_{x \in X} \sum_{y \in Y} d(x, y)$$

where X and Y are two clusters and x is the object in the cluster X and y is the object in the cluster Y.

2.7.2 Partitional Clustering

The partitional clustering produces flat clusters, where every cluster is non-overlapping. To run a partitional clustering algorithm, it is necessary to supply a desire number of clusters K. This is considered as an disadvantage of the partitional clustering compared to the hierarchical clustering. One of the most well-known partitional clustering method is k-means.

2.7.2.1 K-means Clustering

The essential inputs of K-means include the number of clusters k, the seeding clusters and the distance measure. These inputs implies that there are three factors influencing the results of clustering. First, we can determine the number of clusters k by running the diagnostic beforehand. Second, there are various methods to obtain initial centroids. We can generate random seeding centroids but most often, this method leads to a poor result. We can also choose an initial centroid by running an algorithm beforehand, such as the canopy clustering algorithm.

Suppose we have a set n points $(x_1, x_2, x_3, ..., x_n)$ in a multi-dimensional space. The k-means algorithm is to minimise the square of the difference between the subset of points and the mean ϕ_k of a cluster c_k they belong to.

$$K(c_k) = \sum ||x_i - \phi_k||^2$$

And for all cluster k, we have

$$K = \sum_{k=1}^{k} \sum ||x_i - \phi_k||^2$$

The classical partitional clustering technique k-means is easy to implement but it also exposes drawbacks. First, the initial number of cluster heavily affects the quality of clustering. Second, its time complexity is *O(kni)* where k is the number of clusters, n is the number of documents, and i is the number of iterations. Finally, both noise and outliers have great effect to the algorithm.

2.7.2.2 Fuzzy Clustering

The fuzzy clustering technique produces non-hierarchical clusters but clusters can overlap other clusters, making objects possibly belong to more than one cluster. An example of a fuzzy clustering is fuzzy c-means

2.8 Clustering in Information Retrieval

The *cluster hypothesis* states that "*closely associated documents tend to be relevant to the same requests*" [62]. In other words, if we can retrieve a document that is relevant to a search query, we can also retrieve documents in the same cluster with this documents for the search query.

2.8.1 Search Result Clustering

The search result clustering or the query-specific clustering is to apply a clustering technique to a set of search results in order to return to the user coherent groups of results instead of a flat list. This method is useful when the query term carries different senses. For example, the user may search for a types of coffee by the search keyword "*java*" instead of Java programming books. As a result, the clustering technique organises the search results into coherent groups, where each group contains documents of the same topics.

Since the query-specific clustering is only applied to a subset of the corpus, it is faster than static clustering. Nevertheless, since the clustering occurs after building the ranked list of results, it likely increases the retrieving time (the time between search query insertion to result display).

Previous researches show that the query-specific approaches have strong improvements on the document-based ranking search and can produce clusters that fit the search query [86, 31]. All these researches also confirm the correctness of the cluster hypothesis towards the query-specific approach.

2.8.2 Cluster-based Retrieval

Cluster-based retrieval is to compute the similarity between queries and clusters, instead of the documents. For all clusters, there is a ranked list, where the most relevant cluster has the smallest distance between the cluster centroid and the query. According to each cluster, a ranked list of document in the corresponding cluster will be computed. The top documents are considered as the most relevant documents to search queries. Since the number of clusters is significantly smaller than the number of documents, the search is extremely fast. However, it is important to know that there is a possibility that the query will not match any pre-defined cluster.

In previous studies, retrieving by matching between clusters centroids and the query performs poorly when compare to matching with individual documents [27, 66].

2.8.3 Scatter/gather

Scatter/gather technique is a query-specific technique. It was initially developed for document browsing purpose [14]. This technique is inspired by a real life situation, where people often narrow down their searching process in a few steps. Scatter/gather works in the same manner when it first *scatters* the corpus into clusters, then allows the user to select one or more clusters from given clusters. All of the selected clusters are *gathered* and scattered again. Then, the user can choose their narrowed clusters again. This iterative process continues until the user find their desired information. A good example of scatter/gather is Google News[4]

Figure 2.4 is an example of scatter/gather browsing technique. Initial clusters were scattered from 5000 articles of *New York Times News Service* and presented to the user. After the user selected three labelled clusters "Iraq", "oil" and "germany", all articles in three selected clusters were gathered for further clustering. Next, the user picked "Pakistan" and "Africa" in the following step and ultimately they reached smaller stories they felt interesting.

2.9 Clustering Evaluation

There is no inclusive evaluation method for every clustering algorithm. Instead, the evaluation method depends on the domain it is applied. Therefore,

[4]https://news.google.com

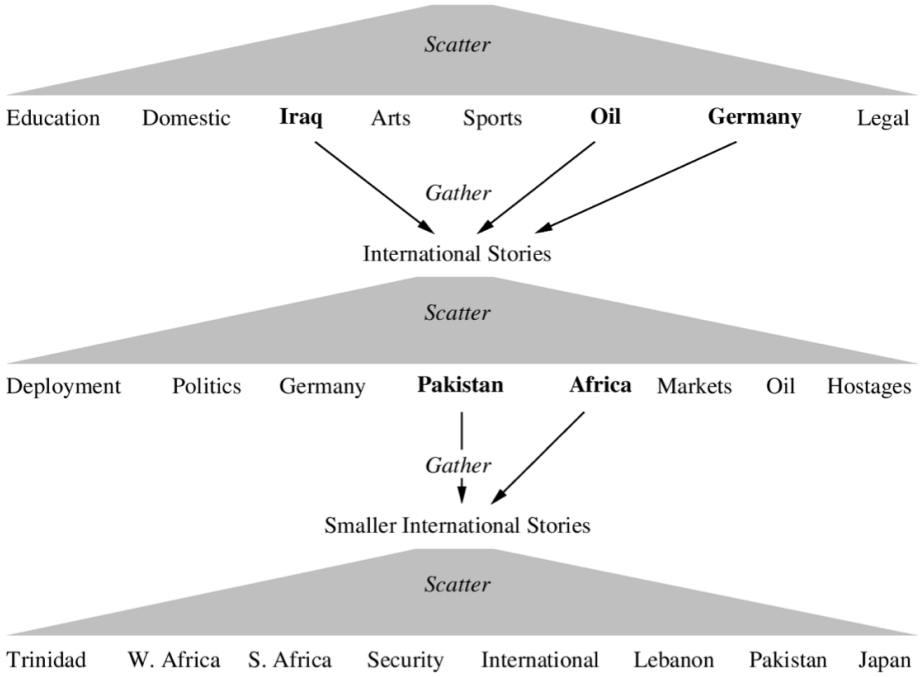


Figure 2.4: An example of scatter/gather [14]

there are different metrics that are applicable to different fields. For example, in IR, f-measure is favorable while in artificial intelligence, the applicable metric could be the mutual information.

Typically, there are external validation measures to evaluate the quality of clustering. *Purity* indicates the percentage of the frequent class are in a given cluster. *Entropy* refers to the distribution of points for each reference class within clusters. The entropy value amplifies when the points diversify, indicating the clustering is not good. The additional validation measure is f-measure and normalized mutual information.

- **Purity** can be obtained by assigning each cluster to the most frequent class of the corresponding cluster, then computing the total correct assigned documents and finally dividing by the total number documents [9]:

$$Purity(\Omega, C) = \frac{1}{N} \sum_{k} max_j |\omega_k \bigcap c_j| \tag{2.1}$$

where $\Omega = \{\omega_1, \omega_2, \omega_3, ..., \omega_K\}$ is the set of clusters and C = $\{c_1, c_2, c_3, ..., c_J\}$ is the set of classes.

- **F-measure** In IR contexts, *precision* and *recall* are both basic measures. While precision is the subset of retrieved information that are related to the request, recall is the subset of relevant information to the request that is retrieved from all relevant information. The f-measure metric is applicable in IR since it takes into account both precision and recall measures.

  For a positive real β, the formula F_β is given as below:

$$F_\beta = (1 + \beta^2)\ x\ \frac{precison\ x\ recall}{\beta^2\ x\ precison\ +\ recall}$$

  When β equals to 1, the precision and recall are equal for corresponding F_1. It is noted that F_1 is considered to be biased [57], there are a number of variation formula, and the most commonly used formulas are F_2 (β=2) and $F_{0.5}$ (β=0.5).

- **Normalized Mutual Information** This metric is applicable when the number of reference clusters diverge from the clusters created by clustering algorithms. For example, we can achieve a good purity by giving a high number of cluster. The mathematical definition is given by [88]:

$$NMI = \frac{\sum_{h,l} n_{h,l} log \frac{n n_{h,l}}{n_h n_l}}{\sqrt{(\sum_h n_h log \frac{n_h}{n})(\sum_l n_l log \frac{n_l}{n})}}$$

3 Literature Review

3.1 The Desktop Metaphor

The desktop metaphor was originally introduced in 1970 [85]. It intrinsically mimics the physical desktop where documents are placed in nested folder structures. The view of a document corresponds to a paper document on the real desktop and miscellaneous tools, such as the calculator, are included in the interface as well. Over the last decades, the desktop metaphor has been heavily employed by popular operating systems (for example, Microsoft Windows or Apple Mac OS X), making it the only well-known working environment to the user.

Interestingly, even though the computer has been through tremendous development in the last decades, the desktop remains almost the same. More importantly, there have been many critical comments towards the usability of the desktop metaphor [59, 60, 46], which include the intensified cognitive load happening to the user imposed by the desktop in basic tasks, such as classification and retrieval [46] or the confusion the user may experience due to the separation of various pieces of information, such as bookmarks, emails, files, etc. [60]. Moreover, putting documents into distinct folders does not make sense when the content of a document spans several topics [46, 19]. More importantly, the organisation of documents depends not only on the document contents but also on the task the user may conduct at a specific

time [18]. Therefore, the imitation of the physical desktop in the form of a nested folder structure is not as advantageous as it may seem. Instead, it may hinder user activities in accessing the document space.

As stated, the classification activity caused by the desktop metaphor intensifies the cognitive load of the user; therefore, it is necessary to support automatic document classification on the desktop [50]. A number of investigations from [50, 32] provide explanations of how the user names the folders and creates the file structure. For example, it is given that the folder name and structure imply a user-defined set of keywords for the documents, which we know as the document metadata. Therefore, instead of forcing the user to complete those excessive activities, the system should define automatic components that are able to exploit the document metadata in order to conduct the classification task for the user [32].

3.2 Document Access on the Desktop

3.2.1 File Hierarchy Navigation

An interesting report about how the user accesses their documents in the system was based on the use of Windows Explorer, the most popular file system visualisation [26]. In about 17% of the retrieval tasks performed by users using their personal hierarchy, the users could not seem to remember the location of the target document and in 75% of the tasks, the users could not recall the file location. These statistics partially demonstrate the profound problem of document access on the desktop.

3.2.2 Document Searching

From the beginning, operating systems have incorporated search engines to help users search for information. There have been some improvements in file browsing in recent updates of common operating systems, such as file tagging, in order to help users organise their information space more efficiently. The user can also use search technologies provided by operating systems, such as Spotlight in Mac OS X, when looking for particular documents. Spotlight in Mac OS X offers search functionality for items such as documents, images and folders. Apart from basic properties like name, date created, date modified, Spotlight can also index the content of certain files. Additionally, Spotlight provides boolean conditions by which the user can search for items with multiple parameters. The user, for example, can find documents that have file names including the keyword “java” as well as using last modified date

(e.g., within 1 week). Another advanced feature that may be familiar to the experienced users is that the user can search using command lines instead of a fancy user interface application like Spotlight. As Mac OS X is Unix-based, computer experts can use built-in scripts to make complex searching queries using the command line. Finally, for each search result, the user can also sort and filter criteria like name, file type or date.

Operating systems have integrated several advanced features into their document browsing, searching and retrieving tools in recent updates; however, none of these new features could create an immense impact that will help people retrieve files more effectively and intuitively. A possible underlying reason is the mimicry of the traditional desktop in the digital space of these file browsers.

3.3 File Hierarchy Visualisation

3.3.1 3D Technique

Since one of the most critical problems with the classical desktop metaphor is the high cognitive load that the user may suffer when navigate through the hierarchical structure of the file browser, some ideas were proposed to replace the existing UI in order to minimise the cognitive load. A common idea is to exploit the perceptual system by replacing the original 2D spatial layouts with the new visualisation of 3D space [63, 43, 64]. The basic motivation behind the idea is the support of spatial memory for human cognition, which means an increased chance of people remembering the organisation of documents in the office. For this reason, the studies expected that spatial memory would be applicable to the virtual space.

Cone Tree [64] visualises the hierarchy information in 3D space where the root of the tree is placed at the top of the space and the children will be displayed along from top to bottom. All nodes at the same level will have the same height and the tree will be displayed in an adjustable ratio that can fit to the window (Figure 3.1). When the user chooses a node, the Cone Tree will rotate in order to bring up the selected node to the front. The rotation is enhanced further by movement animations, enabling the user to perceive the spatial relationships of objects after the rotation. Additionally, Cone Tree exploited three different visualisation techniques to maximise usability. The first technique is to use a fisheye view of the information, while the second technique is to make the 3D depth adjustable in order to reflect the distance from the object to the user. Last, the idealised shadows of the hierarchy reflected to the floor in the image partially impart the structure information

regarding the hierarchy.

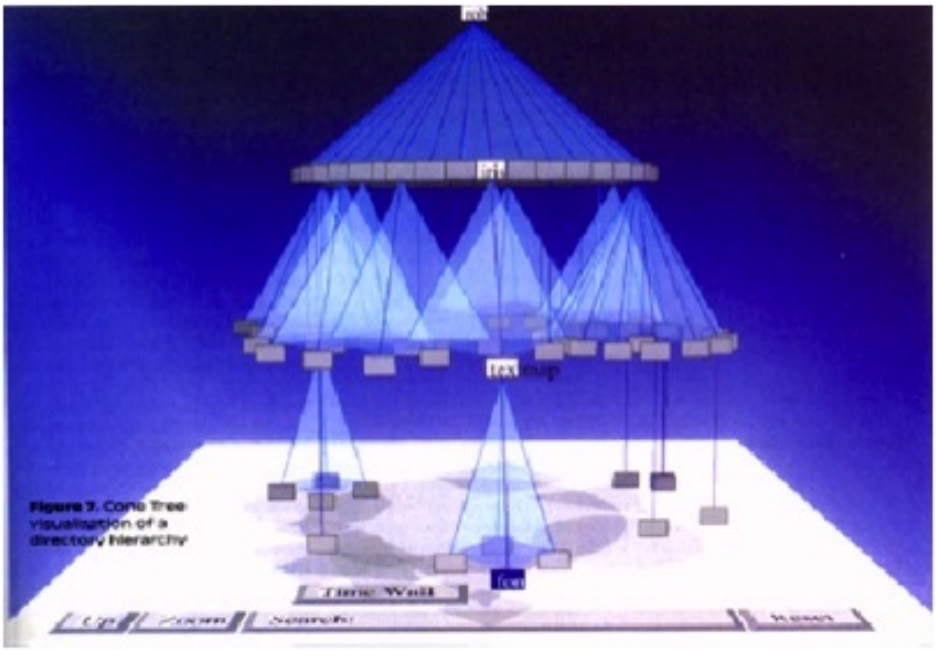


Figure 3.1: The layout of a simple Cone Tree [64]

The Data Mountain [63] prototype, for example, visualises collections of documents by placing icons of documents in 3D space and integrating 2D interaction techniques as general interactions. The flexible dragging interaction enables the user to drag their documents to any place in the virtual space. The user can also perceive the movement of the documents during dragging, enabling the formation of spatial relationships between page objects in the space. Audio is the second modality integrated into the prototype to provide feedback for every user interaction.

All listed examples using 3D visualisation are interesting; however, insight provided by several studies suggests that the performance using interactive 3D visualisation in storage and retrieval tasks is still unclear when compared to the 2D interface. In detail, the completion time for retrieval tasks using a 3D interface is relatively higher than when using a 2D interface [11]. There is no specific reason explaining the difference, but people seem to experience difficulties when visually matching, which partially demonstrates that the 3D interface is less efficient than the 2D interface [12]. Nevertheless, it is interesting to see that the user benefits from the 3D interface in the sense that the 3D interface contributes to building the spatial memory [48].

3.3.2 Focus + Context Technique

The fisheye technique was originally proposed to visualise large hierarchy structures [8, 24]. Typically, the visualisation using the fisheye technique [8] allows two observational perspectives for a single structure simultaneously. The first perspective, the focus, provides the visualisation of information with a high level of detail, using a soaring degree of magnification. The second perspective, in contrast, represents the rest of the information structure at an abstract level, enabling the user to grasp a typical structure of information together with the focus view.

The hyperbolic file browser is a good example of a visualisation that uses the fisheye technique to support viewing a complex hierarchical information structure [45]. The hyperbolic browser initially maps the hierarchy structure in ordinary 2D space onto a hyperbolic plane and then projects it to a circular display (Figure 3.2). The depth of focus is different between nodes. Nodes that are close to the centre of the circle will be magnified to a higher ratio than nodes that are further away. By adjusting the focus depth accordingly, this new presentation is not only capable of displaying tree structures up to a thousand nodes but also preserves smooth navigation. The file browser has certain advantages; however, some problems can also arise. First, compared to 2D or 3D space, the orientation is changeable in the hyperbolic plane when the user traverses the tree. Therefore, though the user can easily understand the overall structure of the document collection with this kind of visualisation, the user may have problems with identifying the location of a particular node in the overall space.

Flip Zooming [2] is another example of the focus + context technique that uses distortion to reduce information. However, unlike other distortion techniques, such as the hyperbolic file browser, both focus and context objects are in a single plane, in which only the focus tile is scaled linearly. The basic element in Flip Zooming is the tile, and the focus tile is placed in the centre with other tiles around it. If the viewer selects a new tile, the focus changes and the selected tile becomes the context. When the hierarchy contains multiple levels, each tile represents a Flip Zooming visualisation containing nested tiles inside. This mechanism implies the interface is capable of having multiple focus levels. While this feature provides easy navigation between documents, it seems to be inappropriate for complex information space.

The focus + context technique in document discovery and retrieval is noticeable but still needs careful consideration. We agree that the focus + context technique is superior to other techniques in terms of providing a comprehensive overview of document entities. Nevertheless, if there is a large number of children, especially near the outer edge, the performance of the

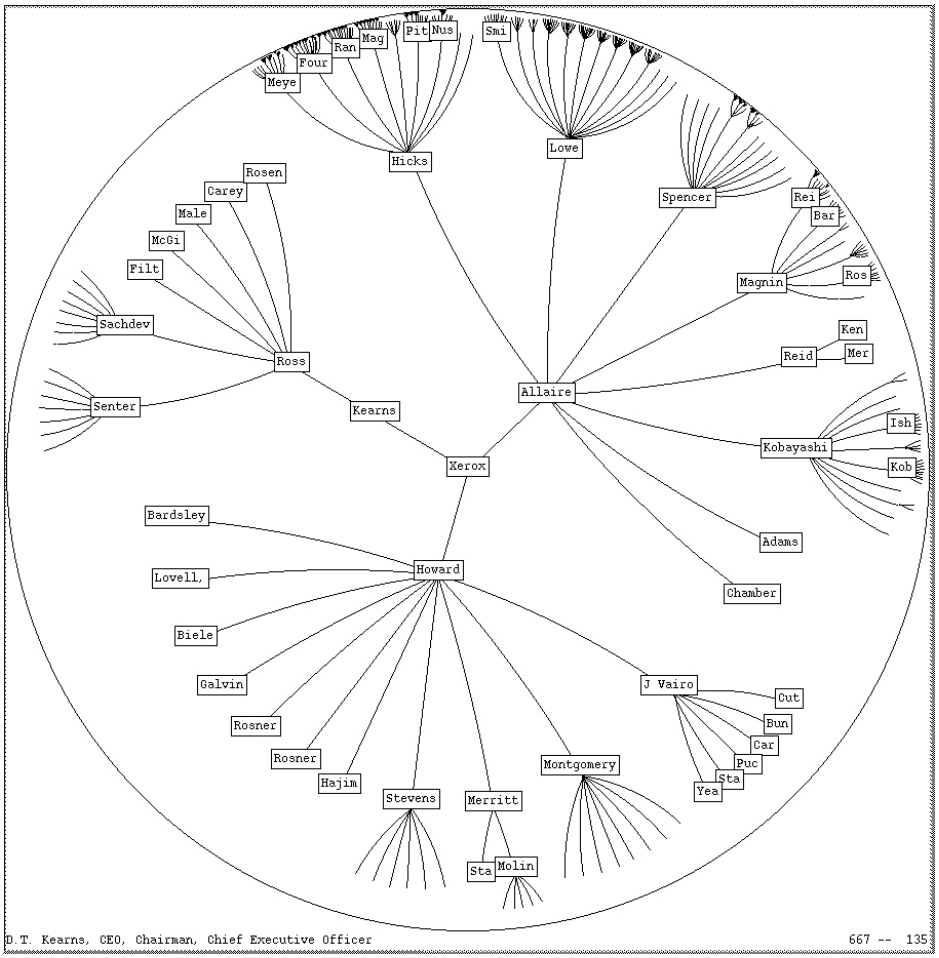


Figure 3.2: An organisation chart using fisheye technique [45]

focus + context interface is weaker when compared to traditional browsers, such as Microsoft Windows Explorer [10]. Also, the focus + context technique seems to be appropriate for demonstrating formulated hierarchical structure but not suitable to indicate random relations where documents are not in the hierarchical structure.

3.3.3 Space-filling Technique

One good example of the space-filling technique is Tree-map [35]. Tree-map visualises the information hierarchy using nested 2D rectangles (Figure 3.3). This kind of visualisation is more favourable than traditional static methods in the sense that it is capable of exploiting 100% of the space. In detail, Tree-map visualises the tree structure using components of inner rectangles

(Figure 3.3). In order to determine the size of the rectangles, Tree-map assigned a weight to each node, indicating the criticality of the corresponding information in the overall structure. While Tree-map is good at depicting information attributes using rectangles, the user reported difficulties in understanding the overall structure of the information space while using Tree-map [78].

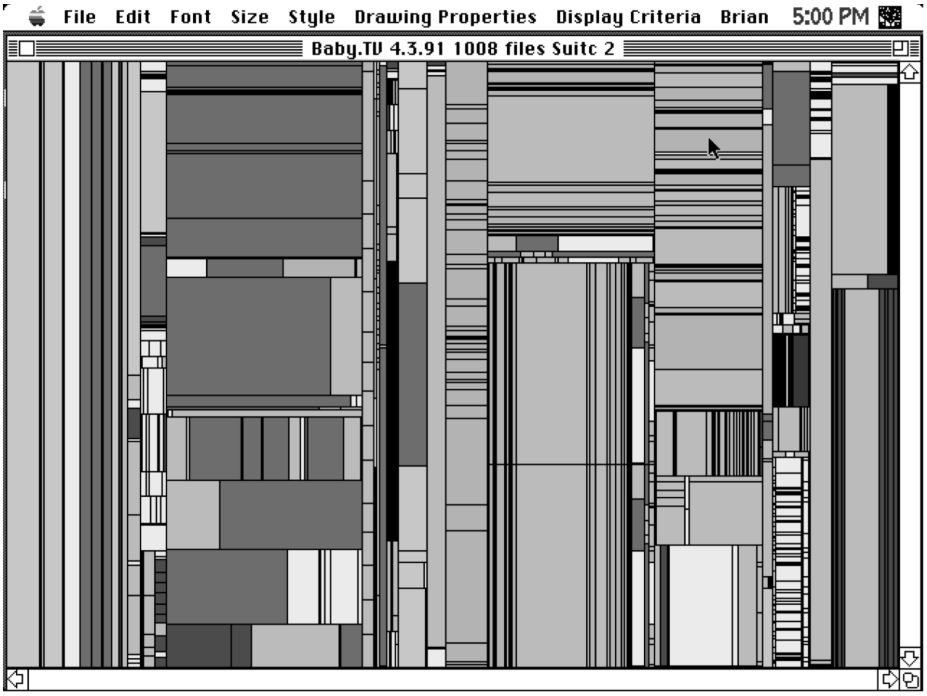


Figure 3.3: The screen snapshot showing a Tree-map of 1000 files [35]

The structure ambiguity of Tree-map visualisation leads to some enhanced interfaces such as Sunburst [79]. The Sunburst tool allegedly conveys both area and structure [78]. Compared to Tree-map, Sunburst is basically the same, as when it also makes use of the space-filling technique. The biggest difference between them is that Sunburst arranges portions of information radically, which is called radical space-filling technique. Interestingly, Sunburst will not occupy 100% of the space as Tree-map does, so the term "space-filling" is partially misnomered in this case. In detail, Sunburst arranges the hierarchy in the following way. The root of the tree is in the centre, whereas the children are arranged further from the centre. The viewer can easily distinguish nodes at different levels by comparing the radius of the discs containing the nodes. Furthermore, the size of the node can be visualised by its radical angle in order to effectively convey the file size to the

viewer.

The concept of the space-filling technique is alluring in the sense that we can exploit the space as much as possible. In addition, we are interested in the overview given by the interface because it depicts the difference between entities, especially the Sunburst interface. Nevertheless, we observe that the space-filling technique is not suitable to document organisation, as it is difficult for the user to understand the hierarchical structure, especially when the tree level is high. Moreoever, the space-filling interface fails to express random trails between entities.

3.4 Techniques for Non-Hierarchy File System

Besides approaches for novel UIs, there are a number of studies working towards adding automated components into the document management system. The purpose of automated components is to help filtering as well as categorising documents into meaningful structures. The use of the clustering technique is one such example, in which the user can obtain more relevant information. The result is significant when documents are organised into relevant categorisation automatically compared to the regular organisation when the user tries to locate a document.

3.4.1 Self-organisation Map

WEBSOM [38] is an example of a document map as a histogram of word categories, allowing the user to explore document contents by either a content-direct search capability or map manipulation. WEBSOM uses SOM [41] algorithm to map documents that have semantic similarities into groups and to organise them onto the document map. The interface of WEBSOM is composed of terms derived from the automatic component and then arranged in the map. The user is able to navigate the map by clicking on the labels or to search to find interesting documents in the map.

3.4.2 Property-based Technique

Placeless Documents [18] states that the concept of the static location of a file system interferes with user needs due to possible problems. First, the document usually refers to several topics, but it can only belong to one place. The static location of the document is more suitable to administrative tasks, such as document backups, rather than document organisation. Specifically, the

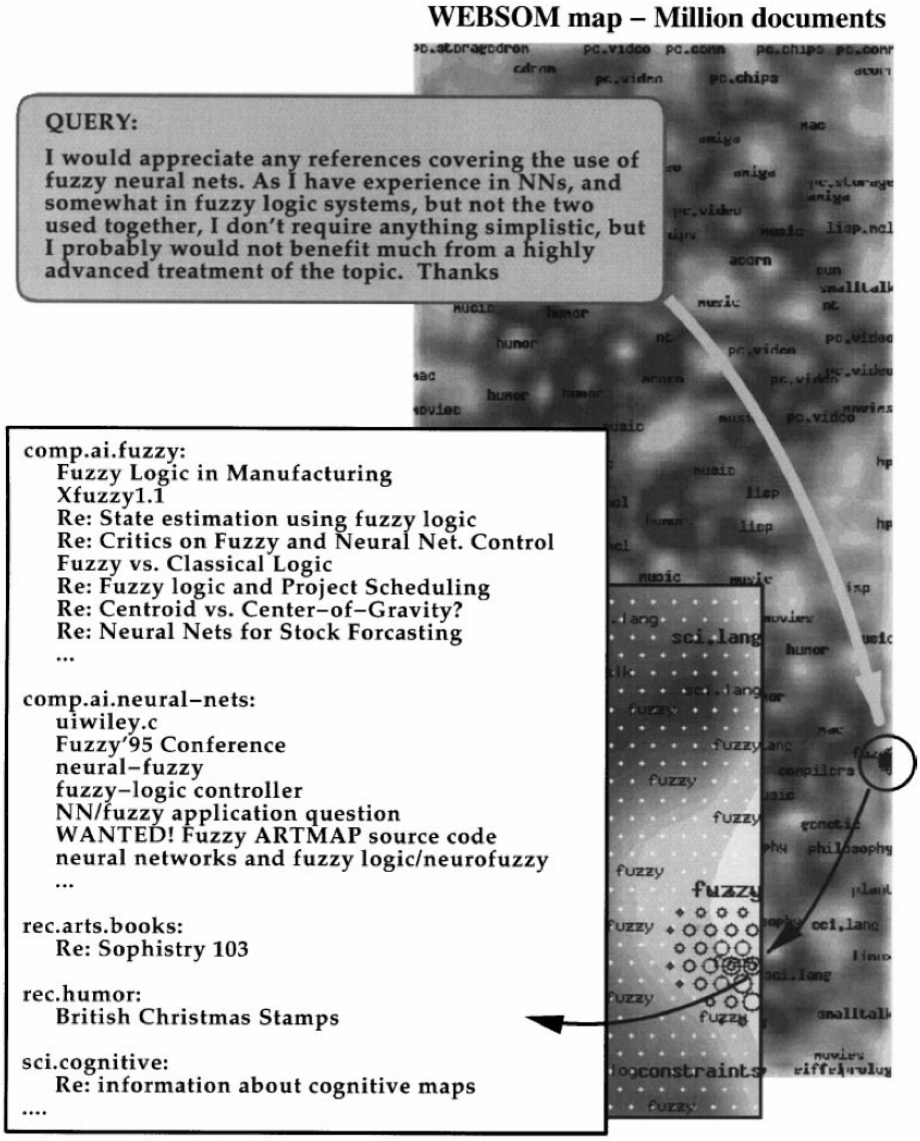


Figure 3.4: An example of a WEBSOM map where the user is able to explore the map using the labels [38]

hierarchy of documents created by a person may not make sense to an other. Therefore, the hierarchy structure hinders sharing activities of documents.

Placeless Documents uses another approach to organise documents. This new approach essentially makes use of the file properties in order to organise the document, making it possible for the document organisation to have multiple facets. Using the properties also allows Placeless Documents to

manage different facets with uniformity of interaction.

The design guidelines of Placeless Documents include three basic features: *uniform interaction* (i.e., document metadata and other arbitrary values associated with the document), *user-specific properties* (i.e., properties associated with the consumer of the document), and *active properties* (i.e., properties to control document behaviour). While the static properties allows the system to group documents, the active properties make it possible to execute management tasks behind the scene. For example, a documents tagged with the "currently in progress" property should be maintained in different revisions.

Another example of the property-based system is DeFiBro [54]. The system harvests the file properties and organises them into a concrete structure using Resource Description Framework (RDF) in order to represent file metadata. All file attributes are then simultaneously pre-processed by removing stop words and punctuation and converting to lower case. For each document, the model queries corresponding metadata values, which will be used to construct a unified metadata index. The indexes are then grouped into synsets using the WordNet lexical database. Additionally, the distance measure algorithm is also applied in order to evaluate the similarities between documents.

3.4.3 Timeline-based Technique

One good example of a file system organised using the timeline technique to visualise the hierarchy is Lifestreams [22]. Lifestreams replaces the hierarchy structure with a novel concept, a stream of files. The file stream arranges documents according to time, where the tail represents documents from the past and the head represents documents from the present (Figure 3.5).

Lifestreams also replaces the concept of the static directory with an additional definition, a sub-stream. A sub-stream can be understood in multiple ways. It is either a virtual view containing documents that satisfy a given query or a temporary directory of documents similar to a static directory in the old file browser. A sub-stream can be defined on the fly whenever the user makes a query. The typical method to create a view is *summary* by which the user invokes a filtering action on the document collection.

The timeline-based technique is indeed a specialised technique that only suits to a relatively small document collection. In case the document size is large, the user has to sift through a long list, which is inefficient. Also the relations between documents are not expressible in this interface. We suggest that the interface should be adopted by other interfaces in order to filtering documents using document metadata such as created date or last modified

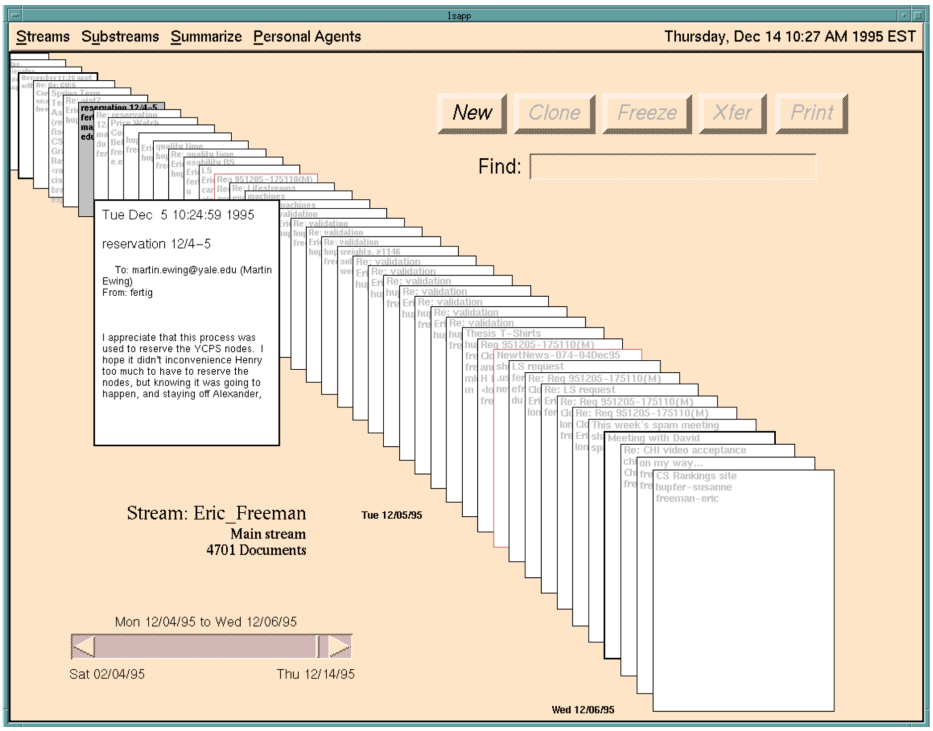


Figure 3.5: A screenshot of the Lifestreams interface [54]

date.

3.4.4 Human Cognition-based Technique

The research concentrates on reducing cognitive load; for example, this includes uses of the spreading activation theory over ontologies [39]. In order to make a decision, people must access information in their brains. The spreading activation theory typically assumes that the human brain stores information in the form of networks of nodes in which distances between nodes are decided by how connected the nodes are (i.e., the link may be shorter for closely related nodes and longer for loosely related nodes). The networks can be in hierarchical form, where concepts are organised in higher categories down to lower categories. The nodes can also be elaborated with private properties and characteristics of each node.

To be more efficient, the human brain stores general properties at higher level nodes and more specific properties at lower level nodes (Figure 3.6). The structure of the hierarchal network plays an important role in decision making. In the given example, people can rapidly verify whether yoghurt is yoghurt, and it would possibly take them longer to verify whether yoghurt is

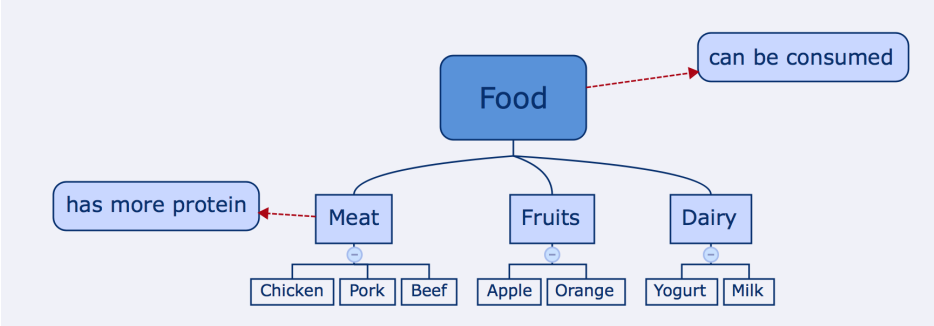


Figure 3.6: An example of a semantic hierarchical network

dairy and even longer to verify whether yoghurt is food. Longer links between nodes result in increased time to verify the links. In contrast, it is different if it takes shorter time to verify whether yoghurt is dairy than it takes to verify whether yoghurt is food. Due to this issue, the network is to be changed from hierarchical to semantic networks in which there are possible direct links between higher level nodes to lower level nodes. This substantial change implies that the nodes would be connected to each other in the network. When people activate a node in the network, they pull out related nodes along with it. The theory is applied over personal ontology, which plays a role as a repository of information for context inference to support user tasks. The idea of [39] is intriguing in the sense that it automatically creates links and weights nodes based on user interactions with nodes rather than requiring users to make hyperlinks between them. Regrettably, the requirement of a personal ontology must be fulfilled to make use of the proposed idea. As mentioned in the paper, it is not easy to create a personal ontology by any means, and as a consequence, it would not feasible to adopt the idea in practice.

3.5 Towards the Memex Vision

In the last section, we briefly reviewed projects that work towards the idea of enhancing document access in digital space. However, all of projects mentioned above still expose limitations. To better understand, we will take a look on the vision proposed by Vannevar Bush in his 1945 paper. In the paper, the Memex described by Bush supports fundamental features: creating links between pages as well as annotating resources and links. However, all the systems we mentioned do not contain any facilities that support linking and annotating resources. Therefore, in this section, we are going to review

studies that take a step towards the Memex vision.

3.5.1 Haystack

Haystack is a novel framework that aims to enhance management of personal information in digital space. The concepts of Haystack are typically based on the flexibility of individual needs when it comes to information management. Haystack emphasises the personlisation of information management, which suggests that the user should take control in deciding which and how objects and their relationships to be stored and represented. In the current systems, the user's information often comprises various types. The information could be a bookmark, email, documents, etc. (Figure 3.7). Each kind of information is managed by individual application, and in order to access a certain kind of information, the user must use the supporting application. This separation of information hinders the effectiveness of the individual since the user usually considers the information to be unified. In Haystack, the separation of information is minimised since Haystack is able to record arbitrary information objects and aggregate them into collections. Haystack offers a compromise between structure and schema in order to pro-

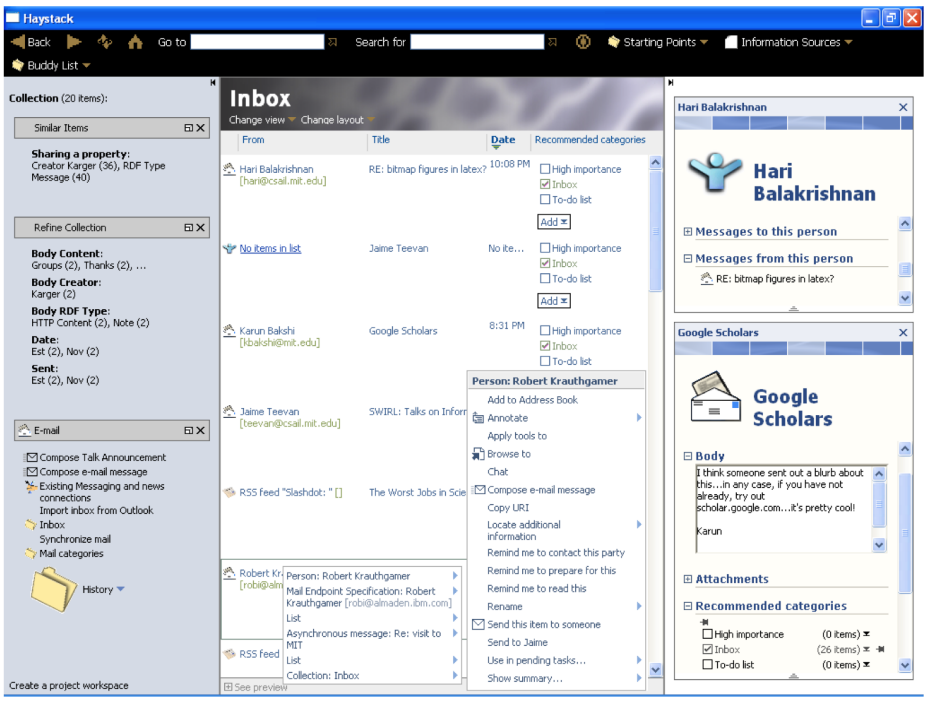


Figure 3.7: An overview of Haystack. The interface illustrates a user's Inbox collection [37]

vide both richness in a data model and flexibility in the UI. The data model structure of Haystack is semistructured, which means that it can offer both the flexibility of a data model by allowing the user to modify the schema and the advantage of a predefined structure like a database-type structure. To create the data model structure, Haystack employs the RDF since the RDF schema is resilient enough to cope with changes in the schema from the user.

To create an interface that is flexible enough to represent aggregations of arbitrary data types, Haystack uses the recursive rendering architecture where the root view only has to know the root object rather than related objects. In addition to the view, Haystack defines *lens* in order to represent groups of attributes for given objects. This facility allows the user to specific their views of different attributes of the objects. For example, the user can create a "help" lens for a certain object and only make it visible when necessary.

Haystack is an excellent example of personal information management system. Using Haystack the user has a lot of flexibility in modifying his information space. Still, we would like to see in the future how Haystack expands its flexibility in terms of truly dynamic links. For example, the user can link parts of an arbitrary email to parts of arbitrary document and annotate the links.

3.5.2 SEMEX

SEMEX targets personal information management by means of creating a search application [7]. Basically, SEMEX makes access to multiple kinds of data possible using a repository of objects and their associations. By means of association, SEMEX defines possible relationships between objects, such as between the email and the sender in address book. Therefore, SEMEX not only focuses on keyword matching search but also identifying associations between objects in order to reveal further relevant search results.

To reconcile references between different kinds of data, SEMEX employs a reconciliation algorithm in its core [17]. Briefly, the algorithm can compare references and make reconciliation decision, and then enrich references based on the results of the propagation of information between reconciliation decisions.

As stated, the searching mechanism in SEMEX exploits references between objects and returns objects that are relevant to keyword-matched search results. To do this, SEMEX defines a *signification score* for objects by weighting the associations between them.

In Figure 3.8 is a screenshot of SEMEX. In the top left and bottom left panels, the user can define the search query for the whole information space

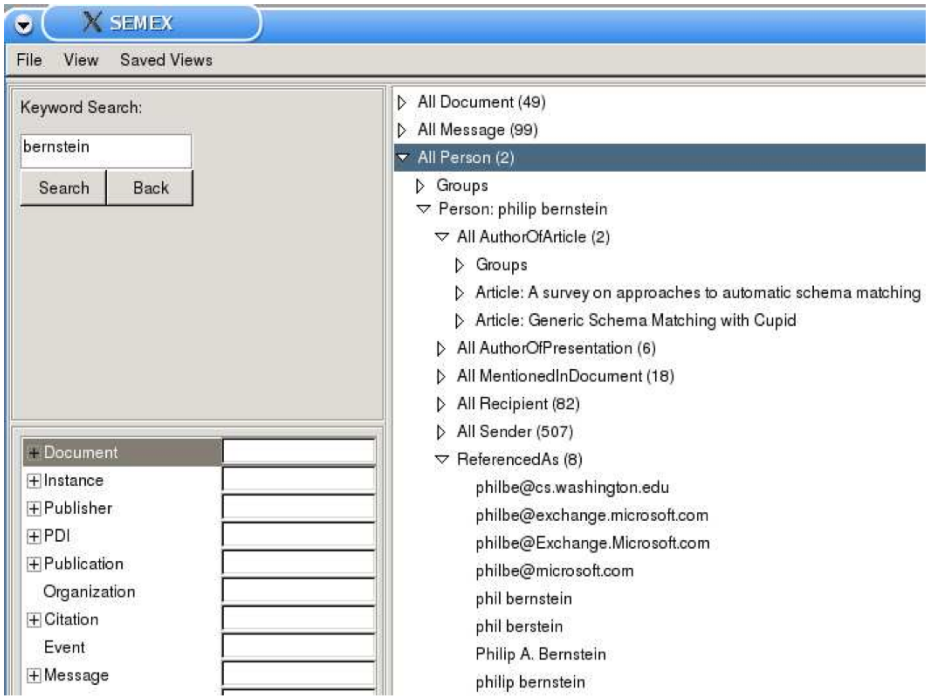


Figure 3.8: An overview of SEMEX interface [7]

or a particular attribute. The research results in the right panel allows the user to explore the search by association.

SEMEX is a comprehensive example of how different kinds of information have references to each other. The reconciliation algorithm of SEMEX is intriguing but also limited in the sense that the user does not have flexibility. If the user would like to manually establish relations between different objects, the system would not allow this. More importantly, while Haystack offers an interactive and meaningful interface, SEMEX simply displays the information in a nested list. This limitation consequently requires the user to shift through a long list when navigating the information. In addition, the search feature of SEMEX does not allow the user to create a complex query or to narrow the search by a particular data type.

3.5.3 iMapping

iMapping is developed on top of the ontology *conceptual data structures* [29]. The formal ontology can also be extended by associating links between items. The file browser resembles the idea of a mind map in which items expand outwards from a central node in a brain-like structure.

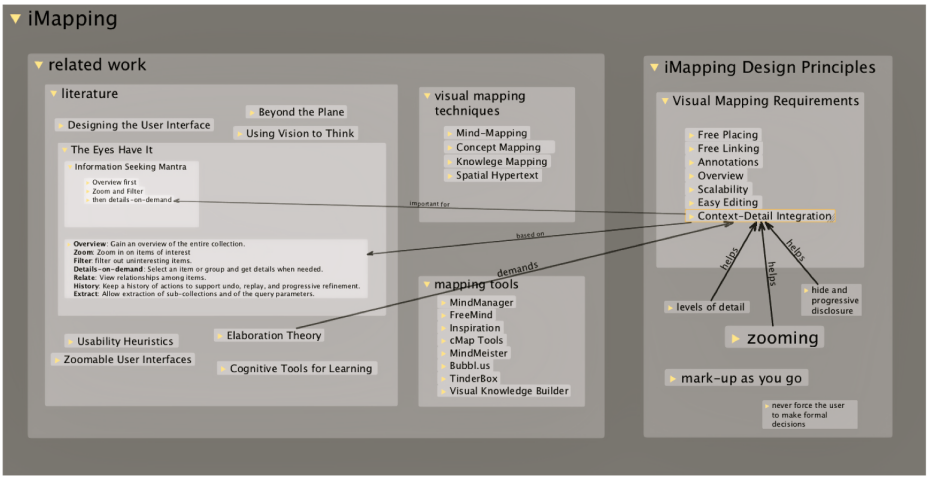


Figure 3.9: An illustration of iMapping with annotated links [29]

Typically, the user can begin with placing resources anywhere in the map. Because iMapping organises resources hierarchically, it allows the user to put resources inside another resource in the same manner as a static directory. By means of nested resources, iMapping creates a view of multiple resources at different levels so that when the user zooms out of the display, the children resources may be hidden from the view, and when the user zooms in, the view will be enhanced with additional details corresponding to the focus context.

Interestingly, whereas the hierarchy structure implies implicit references between resources, iMapping provides additional mechanisms to establish explicit references between them. These mechanisms are labelled and unlabelled links and hyperlinks. While the labelled and unlabelled links are links from one resource to another that are drawn manually by the user, the hyperlink behaves the same as that in the World Wide Web (Figure 3.9).

While the zoomable interface allows the user to zoom, span, and scroll to navigate, the built-in semantic search helps the user to create meaningful queries. The components supporting the semantic search are the *Conceptual Data Structures*, which store data semantically using primitive relations, such as *order, hierarchy, linking*, and *annotation*. To communicate with the front-end interface, there is a java-based CDS-API that supports basic querying and reasoning functionalities.

Even though iMapping allows dynamic links and annotation, the system is confined when it abstracts documents by defining a local resource type. For example if user who uses iMapping wants to exchange information with other systems, the resource must be manually exported to popular document formats, but even doing so, the metadata of the links and the annotations

made would not be retained.

3.5.4 MyLifeBits

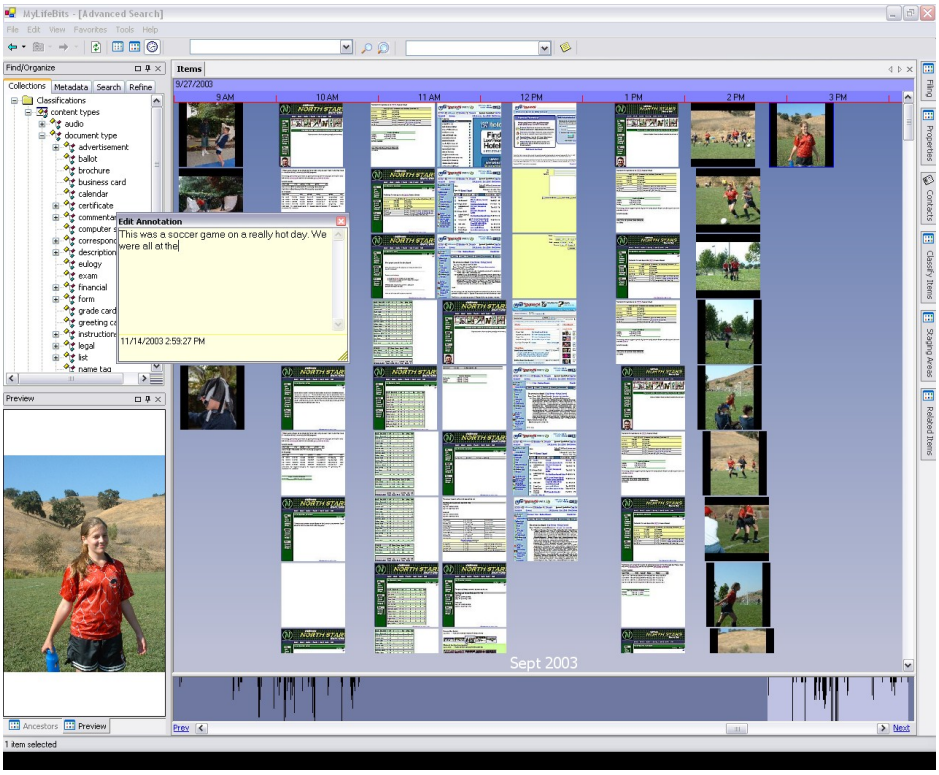


Figure 3.10: An overview of the MyLifeBits application where files are organised into a timeline [25]

The idea of a flexible hypermedia system is also demonstrated by the MyLifeBits project where categorisation no longer exists [25]. Instead, the story is the essential concept of the application. Documents, therefore, are organised by time and space in order to construct a meaningful story for the user.

Besides supporting a keyword matching search, MyLifeBits organises the search results into different views. There are typically four views, including detail, thumbnail, timeline and clustered-time (Figure 3.10). By organising the resources into a specific view, MyLifeBits implicitly forms associations between them. Unfortunately, these associations will be discarded for another search query and not be used in order to enhance further search results. In addition, MyLifeBits allows the user to annotate resources using text and

audio as well. The user can annotate the resource directly or indirectly. The indirect annotation is when the user creates stories using the Interactive Story By Query; stories will be stored as annotations.

To us, the most inspirational idea of MyLifeBits is to organise documents using clustered-time. However, it would be better if MyLifeBits could enhance information retrieval using existing resources such as annotation or possible resources, such as trails between information.

3.5.5 Cross-Document Link Service

The cross-document link service [84] addresses the issue of creating links between documents of different document formats. Using the link service, the user can create and annotate links between parts of documents regardless of the document type. The core component of the link service is based on the RSL hypermedia metamodel [75]. One of the advantages of the RSL metamodel is the flexibility that allows it to support evolving hypermedia systems. With such an extensible core, the link service can adopt different document formats by using data and visual plug-ins.

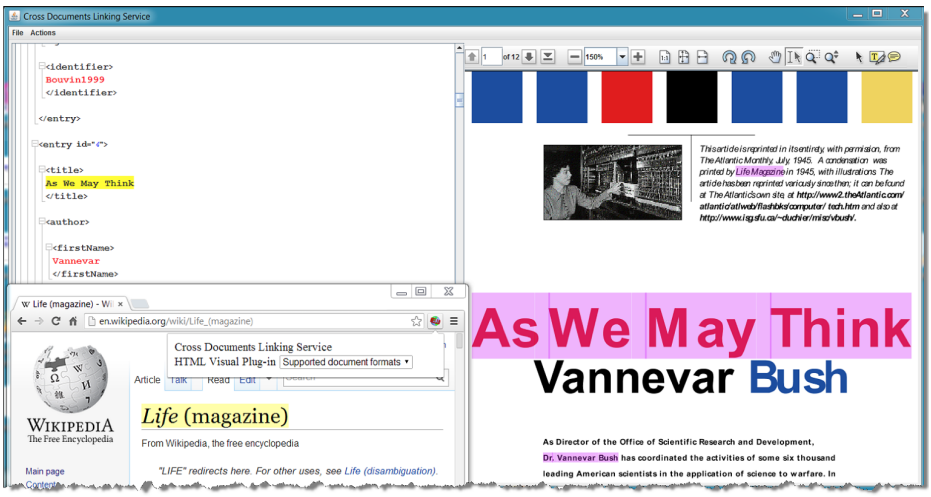


Figure 3.11: Bi-directional links between different document formats created by the link service [84]

Figure 3.11 illustrates bi-directional links between different document formats including PDF, XML and HTML. Whereas the PDF and XML documents are rendered in the prototype, the HTML document is still natively visualised by the web browser. In the figure, the user creates two bi-directional links: one between a PDF and XML document within the prototype and one between a PDF and an HTML document.

Because the explicit link created by the link service can formulate the relationships between documents, we consider it a good resource to enhance document retrieval on the desktop. In the following section, we will outline possible scenarios where the explicit link can contribute to the retrieval task.

4

Enhanced Document Retrieval

4.1 Towards Enhanced Document Retrieval

With the boom of information technology in the last decades, there has been an increasing amount of information for individuals to deal with on a daily basis. People have shifted from paper-based documents to digital documents. Since it is incredibly easy to create, edit and share, the digital document have become extremely popular, allowing individuals to be capable of holding a small library in a laptop. Unfortunately, if the revolution of technology offers the individual many tools to build a digital library, there are only a few instruments that can be used to manage that library.

The first and most common tool every individual can use is the traditional hierarchical desktop file browser. Despite the underlying architectural differences between operating systems, traversing through nested folders is required to locate documents. This method is easy to follow, but previous research shows that it is ineffective and cognitively demanding [26]. Alternatively, the second method is to use the built-in search system to retrieve documents directly from the search query. The search systems, such as Windows Search and Mac OS Finder, allow the user to search based on different criteria. Still, they expose a number of limitations: (i) the user cannot create complex queries, (ii) there is no text snippet extracted from the contents of the document results, the user has little information about the document ex-

cept the file name and (iii) the document result is a flat list as the user must sift through every single item. Other alternative solutions are DeFiBro [54], Cone Tree [64], Dynamic Timelines [43], etc., but they expose a number of shortcomings. They either neglect document content or stick with a traditional hierarchical tree, or are confined within a predefined visualisation. Still, recent research has facilitated the concept of explicit links between documents [84]. As this facility is new, there has been no research exploiting this concept in order to enhance search tasks on the desktop.

All of the issues we have addressed must be resolved. First, it is essential for the user to be able to compose complex queries, such as boolean queries or wildcards. Take SEMEX [7] as a good example. Even though SEMEX provides search functionality to the user, the user can only create simple keyword-based queries. The search result returned by SEMEX is also limited, as it is simply a list without extra information, which we consider a usability problem because the task is more memory-demanding. Therefore, we suggest that for each information object, the new model should provide a rich set of extra information to resolve such problems. Moreover, instead of organising information by static attributes, such as information type, we suggest that the system should organise information dynamically, which could be achieved using clustering techniques. Therefore we would like to promote the use of clustering techniques in our proposed model because it allows us to flexibly create a virtual directory (e.g., document cluster) flexibly. More importantly, the explicit link created by the link service can be used to enhance the search task. For example, we can suggest relevant documents or increase the cluster quality via the explicit links. Finally, the extensibility in terms of multiple visualisations allows the system to provide the user maximum insight regarding the information set.

We have reviewed common issues with previous works and revealed suggestions that can resolve them. Therefore, to make a step forward towards enhanced document retrieval, we would like to formulate a model that (i) is extensible in the sense that it can accept current and future document formats, (ii) offers different kinds of visualisations, (iii) can exploit the concept of explicit links in order to enhance document retrieval task (iv) enhance retrieval tasks by grouping similar documents into groups, (v) can allow the user to create a complex search query.

In Figure 4.1, we illustrate a scenario where we take into account the advanced search query, the document metadata and content, the explicit links between documents and the multiple visualisations. At the beginning, the user can perform their searches using an advanced search form, where they can combine as many fields as they want. There are various fields, such

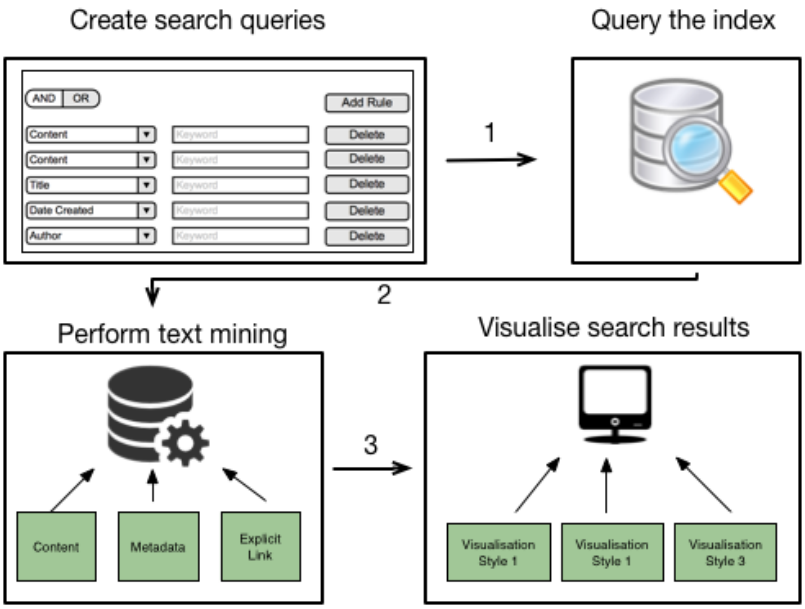


Figure 4.1: An enhanced search scenario

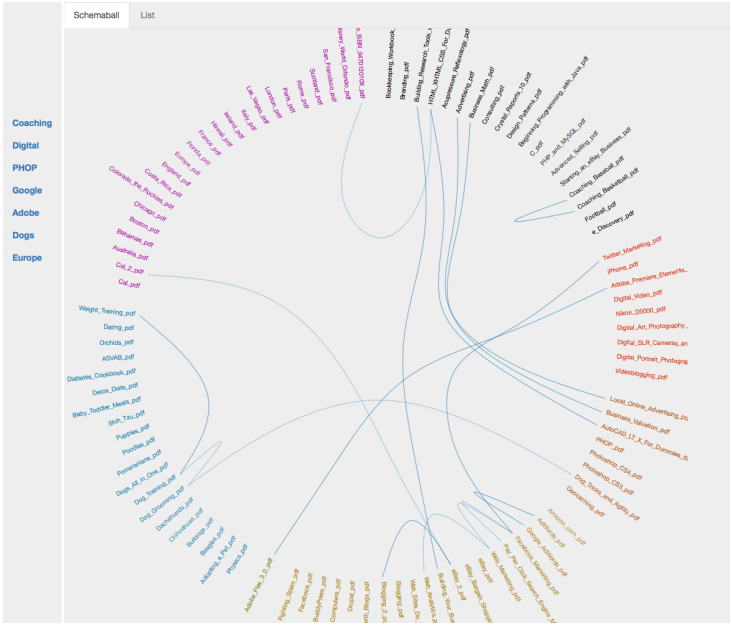


Figure 4.2: A possible visualistation that demonstrates clusters of documents and the explicit links

as the content, the author, the title and the date created. It is also necessary

for the user to use a specific field more than one time. When the system receives the search queries from the user, it will perform the search over the index database, where words of all documents are stored. When the search finishes, it will return a list of matching documents. This document list will be passed to the data mining component. In addition to clustering using the document content and metadata, this component takes into account the explicit links to suggest relevant documents. When the data mining task finishes, the system will visualise the results and display to the user. There will be different visualisation styles that are supportd by the visualisation component and the user can decide the most suitable to use. Figure 4.2 depicts the schemaball as a possible visualisation that demonstrates clusters of documents and the explicit links. In this figure, documents are organised into different clusters on the right panel, whereas the names of clusters are on the left panel. The explicit links are depicted by connecting curves between documents. In the following sections, we elaborate on the requirements for enhanced document retrieval and explain the architecture and all components used to realise the system.

4.2 Requirements

4.2.1 Document Metadata

Metadata is a broad term known by many people as "data about data". According to [20], metadata *"provides a user (human or machine) with a means to discover that the resource exists and how it might be obtained or accessed"*. There are two different approaches towards the issue of metadata: the bibliographic control approach and the data management approach [5]. Regardless of the differences between these two approaches, they both concentrate on using metadata to document information about the resource [20].

We agree that metadata is a useful resource and can influentially contribute to the effectiveness and efficiency of our model. We have investigated scenarios where metadata can be used to enhance the search task. In the traditional hierarchical folder, the files are organised in a static structure. By using metadata, we can provide different perspectives for a unique document repository to the user based on the search query over the metadata. The user, for example, can display data based on the title keywords in combination with the author keywords to create a unique data perspective that suits the user. Moreover, with the support of the boolean and wildcard queries that will be discussed later, the user can specialise their search by combining as many metadata properties as desired to enhance the searches. Second,

the metadata can also be used to suggest relevant documents by text mining techniques. For example, we would create a vector space model from the title property and suggest relevant documents by measuring the distance between the vectors.

4.2.2 Document Content

The document content forms the largest part of a document. It can range from a phrase to a hundred pages; therefore, document content concerns all information of the document. Indeed, if we consider that document metadata is the door to get to know a document, the document content is everything about it and allows us to distinguish a document from others. A good example of how document metadata cause misunderstandings about the information in the document is when we consider documents with the title "Gone with the Wind". The mentioned title may be familiar to many people, because there exists a famous fiction book with the same title. Different from human beings, the search system will not make any assumptions about a document when people search using a query. If people search using "Gone with the Wind", they may get a documentary book about villages destroyed by storm because it seems to make sense if the book is named as such. Therefore, to provide the search system with more in-depth information about a document, besides the document metadata, it is necessary to acquire the textual data of the document in order to provide a better search result.

In our solution, the document content serves two purposes. First, apart from the document metadata, we use the document content to build the inverted index to support full-text search. The second purpose is to use the document content for clustering, but before the clustering process it is necessary to preprocess the document content. During the preprocessing step, we first remove punctuations as well as stop words using a predefined list. Next, we convert all remaining words into lower case and apply a stemming algorithm. After the preprocessing step, we obtain a list of content-bearing terms. The next step is to weight these terms using TF-IDF. At the end of the second step, we obtain a list of vectors that represent our original document collection. The produced vectors will be used later in the clustering process.

While investigating possible uses of the document content in the model, we have also realised the possibility of supporting a synonym-matching search. By means of matching a synonym, we would not only return documents that contain the input keyword but also documents containing keywords that means exactly or nearly the same as the input keyword. If the user, for

example, searches documents using the keyword "*computer*", we can add documents containing the keywords "*calculator*" to the final search result. This feature indeed could be achieved using the lexical database of WordNet [52] by which we would query a list of synonyms for every search keyword and then execute the search with the generated list. Unfortunately, we have realised the synonym matching search has many drawbacks. First, it would be bandwidth and time consuming if we continue sending requests between the local computer and WordNet database for each search. Second, because we built the vector space model using the document content, the dimension of the vector is already large, and using the synonym to enrich the term vocabulary may not be as effective as imagined. We have examined DeFiBro as a reflective example of the document retrieval system that exploits WordNet to enrich the term vocabulary. However, because DeFiBro builds the vector space model using the keywords extracted from the document metadata, the term vocabulary of DeFiBro is extremely small compared to ours. That is why WordNet is considerably useful to DeFiBro.

4.2.3 Using Clustering

Clustering is the method that is used to group similar information into categories. It has proven effective in reducing the time and effort of search tasks [87]. Vivisimo [42] is a famous example that illustrates the application of clustering for a Web-search engine. In desktop browsing, clustering is appealing in the sense that instead of forcing the user to scroll through a long list of documents, it offers an opportunity that allows the user to minimise the scale of search results by organising documents into different subsets. Thus, the user can have a broad view of the search results, which is beneficial in terms of reducing searching time in retrieving and finding documents. More importantly, clustering helps the user avoid typical scenarios, such as when the user encounters documents with the same name, they can be easily distinguished by selecting the category name. Another scenario is when the user search for a common term, such as "twilight", and obtain many documents of the same name in different categories.

Even though clustering is beneficial for desktop file browsing in the mentioned scenarios, it is still an unimplemented feature of search systems, such as Windows and Mac OS. The user usually has no information about documents returned by the search systems, except the document filename. When the user encounters a large set of documents, it would take an enormous amount of time to examine the search list. In Figure 4.3, we demonstrate the need for document clustering for the search results in the desktop by using the keyword "java" to perform the search. For a common keyword, we

encountered a large number of documents whose topics ranges from learning Java, Windows Form to a job resume, which creates difficulty when trying to locate a specific document.

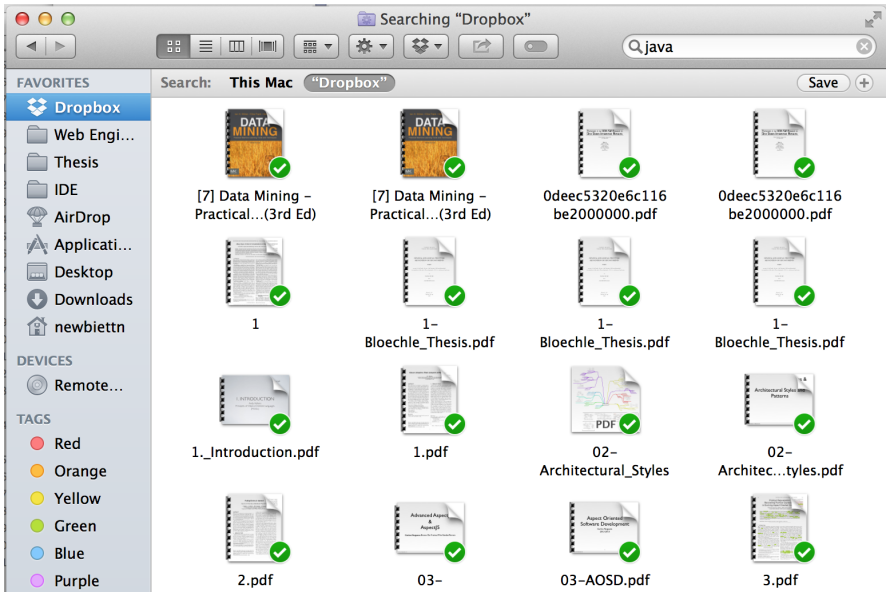


Figure 4.3: An example of uncategorised documents of a search result returned by Finder in Mac OS

Instead of leaving the user with a flat list of document results, grouping documents into categories using clustering promises a better solution. Figure 4.4 illustrates how similar documents can be grouped and visualised. In the figure, documents are no longer in a flat list. Instead, document items in the same group are placed next to each other and have the same representational colour. Thus, there are a large number of documents though, the user is still able to have a good overview of the document result.

To provide clustering to the search result, we investigate the feasibility of dynamic clustering or query-specific clustering. Traditionally, cluster analysis was applied to the information retrieval system in order to increase the effectiveness and efficiency of the system [67]. Since then, it is common to use the clustering method statically; the use of clustering method is applicable to a stable set of document collection. Besides the static application, an exceptional use of clustering is to apply the clustering method dynamically based on a given input (e.g., user query), which implies that cluster analysis is restricted to a subset of document collection that is pertaining to the input. The novel use of cluster analysis is alluring in the sense that if we take a look again at the Cluster Hypothesis, we can assume that query-specific

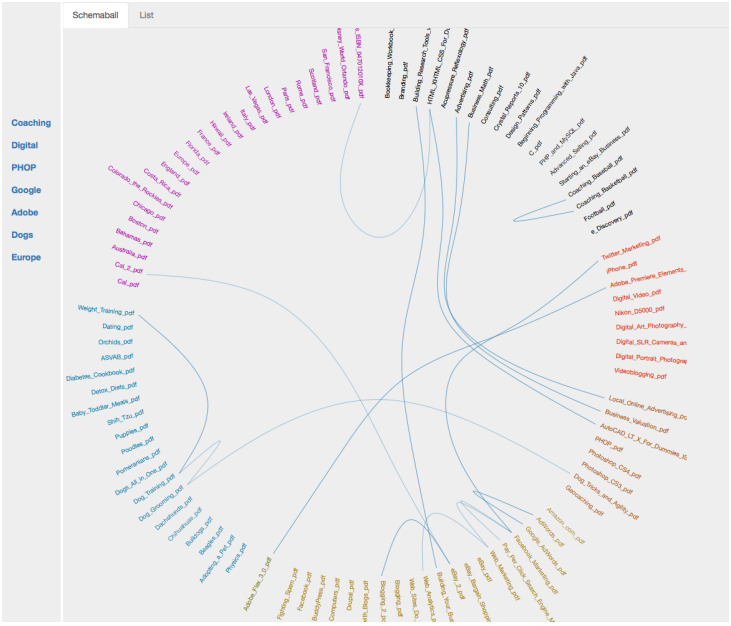


Figure 4.4: An example of visualising clustering of documents dynamically

clustering can return better results when categorising documents.

The case is alluring but so far, there are only a few examples of investigation of the effectiveness of query-specific clustering [58, 31]. Additionally, though we can expect non-static cluster analysis would yield a better categorisation, in return, we can assume that the application that uses this method might be more time-consuming than the traditional method since the clustering process is conducted during the running time.

4.2.4 An Extensible Architecture

This section describes our motivations for an extensible architecture that can cope with a variety of document formats as well as support multiple visualisations. We first explain our investigation of the diversity of documents and how it motivates us to build a system that not only read many kinds of formats but also accepts future document formats. Second, we demonstrate that the system should not be confined by any visualisation style. Instead, it should be flexible in the sense that new visualisations can be integrated into existing components without hassle.

4.2.4.1 The Multitude of Document Formats

In the beginning, the original purpose of the electronic document was to replace the paper-based document so that it is possible for us to create, edit and then print it out. Later, people started to think about the idea of document distribution when the computer network had achieved important steps in development. Instead of printing documents and sending them in person, people have distributed documents digitally. This adjustment has brought convenience to our daily lives. However, it has created a fundamental problem that we still cannot solve, the incompatibility of document formats.

To understand the problem of document format incompatibility, we compare the variety of document formats to the diversity of languages. Typically there are hundreds of languages in the world, making it impossible for people to communicate if they do not know the other languages. In the same manner, due to particular purposes, there is a large number of document formats in use. The most popular document formats include PDF, XML, DOC, XLM, and if we meticulously look at one of the most common file format today, the PDF file, there are eight versions of the PDF format in total [30]. Nevertheless, a document retrieval system must be able to read multiple document formats in order to extract the textual content as well as metadata for indexing. Still, it would not be sufficient if we build a search system that solely focuses on existing document formats because the system would become obsolete as soon as there are new document formats. For the given reasons, the future search system should not be confined by any concrete architecture, instead it should be extensible in the sense that we can increase the number of document formats it could read by adding new libraries.

4.2.4.2 Support Multiple Visualisations

In the same manner as the ability to support existing and future document formats, we make a step forward towards the extensibility of visualisation for three distinct reasons. First, every visualisation schema has its strengths and weaknesses. Therefore, being confined to any particular visualisation schema will limit our model. Second, the flexibility in supporting multiple visualisations offers us additional opportunities to widen our investigation on how different visualisation styles affect the document retrieval task. Third, being extensible to accept different visualisation implies that the user can switch between various visualisation schemas; therefore, the user can would have more than one choice in choosing an appropriate visualisation style

4.2.5 Explicit Links of Documents

Aside from textual content and metadata, explicit links produced by the link service are an excellent resource we can exploit to enhance the search results. In this section, we analyse a number of possibilities where the explicit link facility is applicable to our application.

4.2.5.1 Introduce Relevant Documents to the Search Result

Since the ultimate goal of the link service is to allow the user to create trails between documents, we can assume that when the user creates a link between document A and document B, both documents are related to each other to some extent. Therefore, for a given input query, if the document A belongs to the search results, it is likely that document B may be of interest to the user. Within the scope of this application, we exploit this assumption to enhance the search results. In the use of the explicit links in our application, we do not evaluate the relevant degree of the link between documents A and B. Instead, if documents A and B are linked to each other via an explicit link and document A contains the search result keyword, we will add document B to the search results (Figure 4.5).

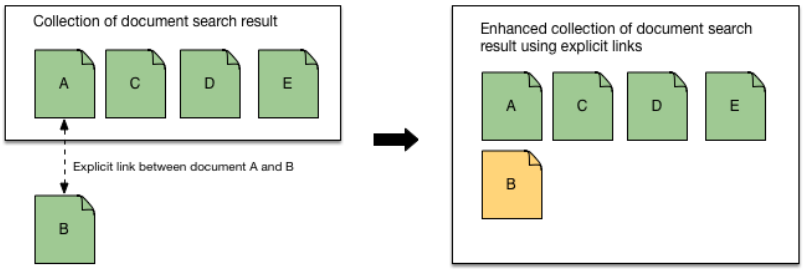


Figure 4.5: Since A and B are documented as linked via the explicit link, we add document B to the search result

4.2.5.2 Increase Clustering Quality

According to the Cluster Hypothesis, documents that are similar to each other tend to belong to the same clusters. If documents A and B have a link between them, we can assume that documents A and B are relevant to each other; therefore, it is more likely that they may belong to a cluster. Nevertheless, the case would be more extreme if we realise that the similarity is only applicable to a particular perspective where the user establishes links between documents A and B. Unsupervised clustering algorithms do not

follow the same manner, where they group documents based on the document vector similarity. The vector is constructed from textual values, including content as well as metadata, whereas the metadata from the link service is not textual data that can be used to model the vector space. Due to this limitation, we leave the case of using explicit links to enhance the clustering quality for future research, and therefore do not exploit explicit links for clustering enhancement.

4.2.5.3 Stimulate Creating Explicit Links for the User

Besides using explicit links to enhance search-related activities, we would like to see how the application may encourage the user to create further explicit trails. When the user grasps a comprehensive overview of the categorised documents and is able to see how the documents are related to each other, there might be interest in adding links between them.

4.2.6 Enhanced Search Features

We have addressed that the common issue of built-in file search systems in both Mac OS X and Windows is the constraint possibility of both search method and display. For instance, using the Mac OS Finder, all we can do to hone our search is to combine criteria in multiple fields, including document type, file name, contents, created date and last opened date. There are many possibilities the search system is missing.

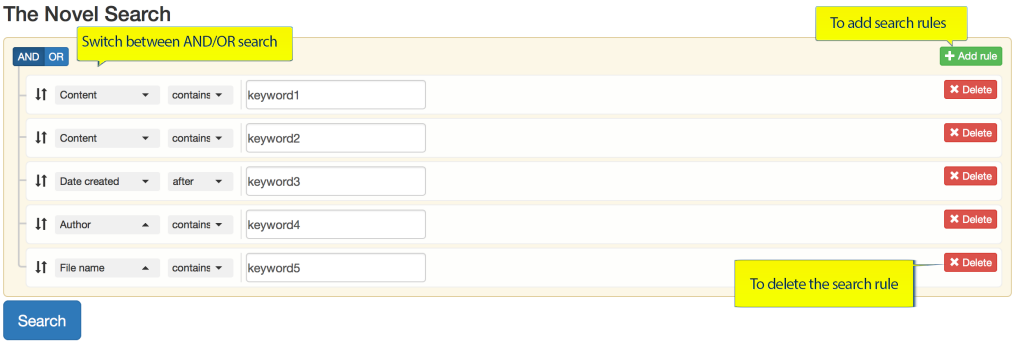


Figure 4.6: An example of an enhanced search form

4.2.6.1 Boolean Query

In Mac Finder, the combination of various criteria is set with the logical operation AND by default, which means that we are unable to create a

flexible query with logical operation OR or both. In Figure 4.7, we illustrate an example of a search in Mac OS using Finder. The example is a search that queries PDF documents containing "java" as the keyword, created within 100 days, and last modified within 50 days. By default, Mac Finder only offers the conjunction search, which means we could not search for documents that satisfy either one of the three conditions. In contrast, we illustrate an example of an enhanced search form (Figure 4.6), in which the user can combine as many fields as they want. The user can also search using a specific field more than one time. There exist various fields, such as the content, title, author, date created. The user can use either the disjunctive search or the conjunctive search to formulate the search logic.

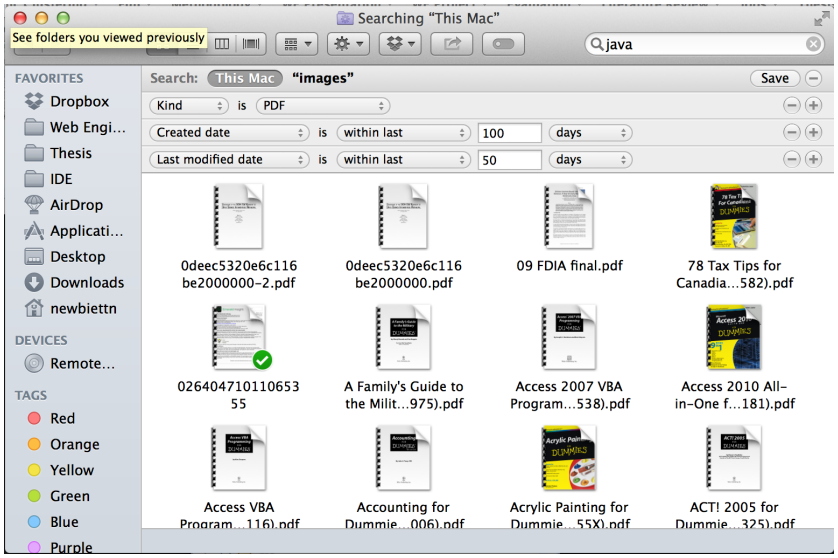


Figure 4.7: An example of a conjunction search in Mac OS Finder

4.2.6.2 Wildcard

Some applications, such as SQL or MS-DOS, heavily use the concept of the wildcard in the search query because it offers a tremendous benefit to the flexibility of the search query. For example, people who are looking for documents whose title contains the keywords "compute", "computer" and "computing" would apply the wildcard *comput**.

We mentioned scenarios that are convenient to the user but are unsupported by built-in visual search systems, including Windows Explorer and Mac OS Finder (the user can use the wildcard in the command line interface, such as Windows MS-DOS or Mac OS Terminal). To overcome such

Query	Seach for documents
english	Contain the term *english*
english france	Contain the term *english* or *france*, or both
english AND france	Contain both *english* and *france*
title:english	Contain *english* in title field
title: english -subject:france	Contain *english* in the title field and do not have *france* in the subject field
*english**	Contain terms that begin with *english*
english~	Contain terms that are close to the word *english*, for example, *england*
publisheddate: [1/1/2014 to 1/1/2015]	Have published date values between 01/01/2014 and 01/01/2015

Table 4.1: Example of query expressions that can be handled [30]

limitations, we supply our search system with additional flexibility in terms of the search query composition. To achieve such flexibility, a dynamic composition of the field query is illustrated in Figure 4.8. In the beginning, there is a query with multiple fields. Query parser then handles the search query. The duty of query parser is to analyse the given query, output it as a form of distinct combinations of field name and field value. For each combination, query parser will match it with a corresponding field name and use the given field value to search documents based on the matching field. Table 4.1 illustrates some examples of the search query the query parser supports.

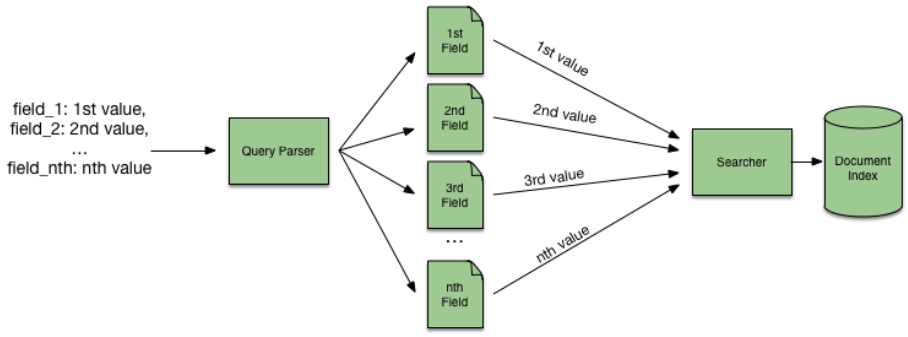


Figure 4.8: The conceptual model of an advanced query

4.3 Architecture

This section describes the architecture of our proposed model that satisfies the requirements we have specified. We first focus on explaining how it is possible for the system to accept different document formats as well as different clustering methods. Then, we describe the procedure by which we index the document collection and construct the vector repository.

4.3.1 General Architecture

In general, the architecture consists of three distinct modules (Figure 4.9). The first module is the searching module where the full-text search is performed. The second module is the clustering module where we proceed with the clustering of documents. The third module will collect the search results as well as the output from clustering to formulate the search results.

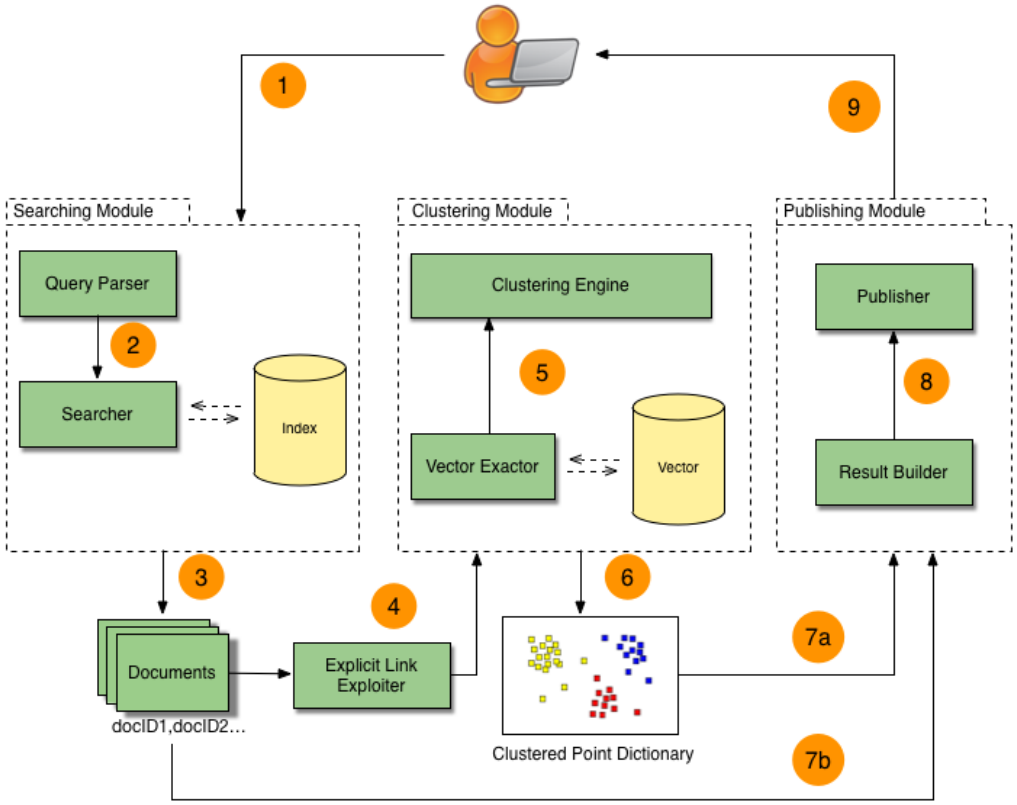


Figure 4.9: The conceptual architecture

In the beginning, the user inputs a search query using the search interface.

The search interface makes a corresponding request to the system. The request made by the user is handled by the searching module where the query is parsed and searched over the index. When searching completes, it returns a list of matching documents based on the given query. After that, the search result is passed to the explicit link exploiter, where relevant documents will be retrieved via explicit links and added to the list. The enhanced search result is then passed to the clustering module. In this module, the vector exactor returns matching vectors based on the list from the vector repository. The clustering engine reads the matching vectors and performs the text mining process. When it finishes, it returns the clustered point dictionary that describes the final cluster result. Both search results and the clustered point dictionary are passed to the publishing module where the search result is finally formulated into categorised structured and ready to be published for further visualisations on the client side.

4.3.2 Indexing

In this section, we explain in detail the process of building the index with the extensible parser component. Together with the parser is a representation we define to exhibit a document entity within our application. We name it the unit document. In Figure 4.10, the unit document will be consumed by index writer to produce the index.

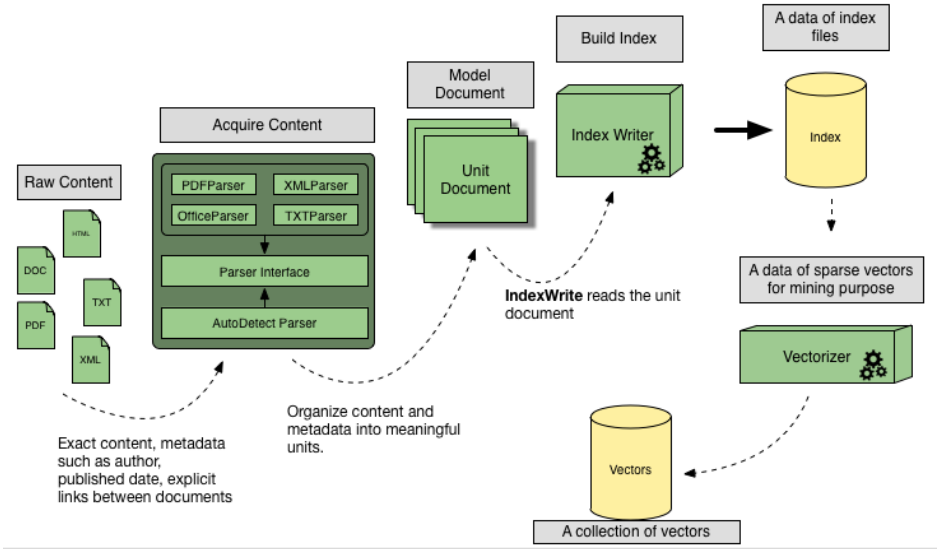


Figure 4.10: The process of building the index and vector from documents

4.3.2.1 Information Extraction

Typically, there are many kinds of document formats, and the very first step is to pull out the information from given documents. In the model, we use the extensible parser to consume documents. In return, the parser will produce text fragments from both the metadata and content. The text fragment is modelled using the unit document in the next step.

The conceptual model of our the extensible parser is illustrated in Figure 4.11. The parser model is extensible in the sense that it is possible to add a new parser to the parser repository. Together with the parser repository, there are two additional repositories, including the language repository and MIME type repository. The language repository is employed by the language detector in order to detect the language of document automatically. Similarly, the MIME type repository is employed by the MIME type detector in order to identify the MIME type of the document. When a new document is passed to the interface, the exactor facade will automatically detect the language and the MIME type of document and correspondingly invoke the appropriate parser in order to pull out the content and metadata of a document.

In general, the model offers a number of features:

- **A general abstract class for parsing**
  Even though there are a large number of document types, the model simplifies the process using a general parsing facade that includes all parsers, language detection, and MIME detection.

- **MIME database and detection**
  The classification of document formats is stored within a single database, and this database is extensible enough for further MIME types. In addition to the MIME database, the MIME type detector class correspondingly detects the MIME type of the document by referencing the MIME repository.

- **Language database and detection**
  Since documents could be composed in different languages, the language repository allows adding new languages for further language detection.

- **Extract metadata dynamically**
  When parsing the document, not all metadata will be extracted. Instead, we are able to specify which kinds of metadata we need.

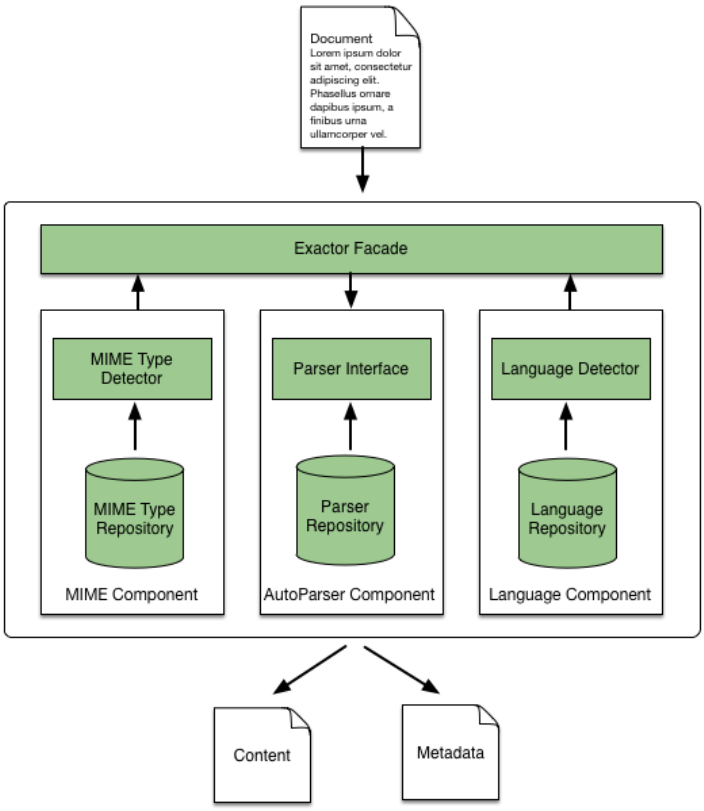


Figure 4.11: The conceptual model of the extensible parser component

4.3.2.2 Document Representation

As soon as the content and metadata from the documents are pulled out, we need to index all of the information for a further full-text search. Before proceeding to the indexing step, we first need to translate and composite all of the text fragment we extracted by using a representational model that the index writer can digest. Since every digital document contains not only the plain content but also extra information, such as author, title, etc., we define a model that represents a *document entity* with all additional fields. In the general model, we define it as the unit document or the *document model*. Therefore, for every document model, there will be a number of predefined fields that match actual fields of the document entity. These fields include *author, title, file URI, filename, published date*.

Moreover, for each field, there are several possibilities. The field is either converted to the inverted index (i.e., indexed fields), or stored (i.e., stored fields), or both (i.e., indexed and stored fields). While indexed fields are fields

whose inputs are plain text and the output are inverted indexes, stored fields take the inputs as plain text and store it permanently for further retrieving purposes. Indeed, stored fields consume more space than indexed fields, but in contrast, they allow original text retrieval when the indexed fields go through additional steps when we want to recover the original forms of the indexed fields.

As discussed above, the metadata of documents are fields that are suitable for storage since they are small in size and we can easily retrieve them later. In contrast, the document content should be indexed, not stored because it is space-consuming. More importantly, we need to analyse the document content to remove stop words or punctuation. The analysing step not only helps reducing the field size but also contributes to further text mining in the following steps.

In general, we define our document model with fields as in Table 4.2.

Field name	Indexed	Stored	Analysed
Content	YES	NO	YES
Author	YES	YES	NO
Title	YES	YES	NO
Filename	NO	YES	NO
Published date	NO	YES	NO
File URI	NO	YES	NO

Table 4.2: Field types of the document model

4.3.3 Extensible Clustering Engine

This section describes our conceptual model of the clustering engine that is centred around the extensibility feature. Figure 4.12 illustrates the component model where the clustering facade abstracts individual clustering components, such as k-means, fuzzy k-means, from the rest of the application. Therefore, our choice of clustering algorithms is not restricted to any specific method. Instead, we can enrich the algorithm library by further clustering methods in the future.

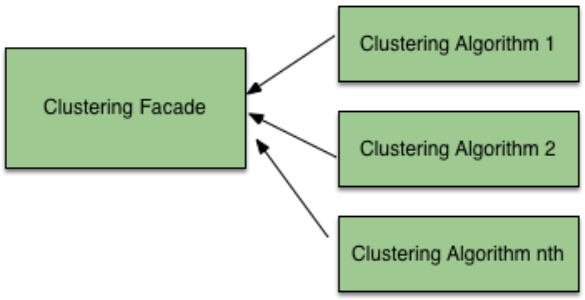


Figure 4.12: The conceptual model of the extensible clustering component

5 Implementation

5.1 Objectives

Typically, there are four propelling objectives of the implementation. The first and foremost is to develop a prototype as a proof of concept for the proposed model. The second objective is to prove our enhanced search features can assist the user in retrieval tasks. In addition, the implementation is a manifestation of using explicit links visioned by Bush to improve retrieval of documents on the desktop. Finally, we illustrates the extensibility of our architecture with two different visualisations, the schemaball and the ordinary list.

5.2 Technologies

5.2.1 Document Parsing with Apache Tika

Apache Tika is a de factor open source document parsing library that supports exacting texts from a large number of document formats. It was initially proposed by Doug Cutting in Apache Software Foundation (ASF). Generally, Apache Tika offers special features:

- **A general abstract class for parsing**

There are a large number of document formats though, Tika provides a general parsing facade that includes all parser classes, language detection and MIME detection. Its goal is to offer simplicity.

- **MIME database and detection**
  The classification of document formats is stored within a single database and this database is extensible enough for further MIME types. In addition to the MIME database is the MIME detection class that correspondingly detect the MIME type of the document by querying the MIME database.

- **Language database and detection**
  Since documents could be composed in different languages, the language database allows adding new languages for further language detection.

- **Low memory usage**
  The text is parsed incrementally into SAX-based XHTML events, allowing the low memory consumption.

- **Extract metadata dynamically**
  Apache Tika does not exact the metadata using a fixed structure. Instead, it uses a user-defined template when extracting the metadata. For example, we can decide which types of metadata Apache Tika needs to extract, such as the author, the title, and which types of metadata have to be discarded, such as the file size.

5.2.2 Search with Apache Lucene

Apache Lucene is a de facto open source library targeting search and search-related tasks such as indexing or querying. Lucene uses the inverted index data structure to index documents. The indexing strategy of Lucene is different from others when it creates segments and merges them, rather than uses a single index. Typically, an index segment consists of a dictionary index, a term dictionary and a posting dictionary.

Figure 5.1 illustrates an example of an index, where the merge factor equals three. In the example, there are 14 documents in the collection. Lucene repeatedly creates a segment for a document and periodically merges a group of 3 documents. The process is similar when Lucene keeps merging a group of 3 segments until there is no more segment to merge. After merging, all greyed segments will be removed and in total, Lucene merged 5 times

after the indexing finishes. The approach of merging and deleting segments is considerably useful as the document collection does not change frequently.

More importantly, the segment will never be modified. Instead, Lucene creates new segments when the document collection changes and later, it merges segments into new ones and deletes the old segments. This strategy ensures there is no conflict between reading writing indexes. Furthermore, this strategy also allows Lucene to avoid complex B-trees to store segments. Instead, all segments will be stored in flat files.

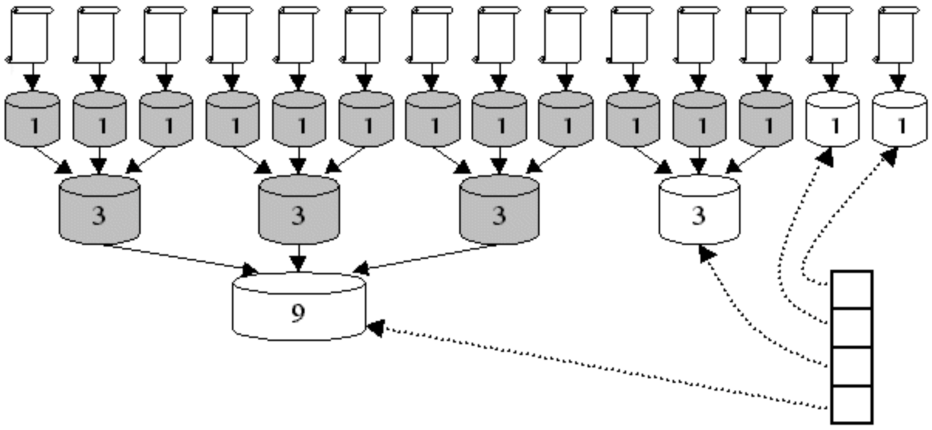


Figure 5.1: The index diagram with the merge factor b = 3

5.2.3 Text Mining with Apache Mahout

Apache Mahout is an open source machine learning framework created by Apache Software Foundation[1]. The idea of Mahout was originally proposed in 2008 in the project Apache Lucene. Since machine learning is a general term, we have to distinguish which areas the library covers. At the moment, Mahout provides frameworks for the following areas.

- **Clustering**
  The purpose of clustering techniques is to group similar items into categories. At the moment, the framework has clustering algorithms, including k-means, fuzzy k-means, streaming k-means and spectral clustering.

- **Collaborative filtering**
  Collaborative filtering is a generalisation process that introduces the

[1]http://www.apache.org

idea "I like what you like" in the social media context. The framework includes typical filtering techniques such as user-based collaborative filtering, item-based collaborative filtering.

- **Classification**
  Classification is the process of assigning unlabelled items into predefined categories. The framework provides several classification algorithms including naive bayes, random forest, logistic regression.

5.2.4 RESTful Web Services

REST or *Representational State Transfer* is an architecture style that relies on the HTTP protocol. The idea of REST is to offer simpler web services to replace complicated mechanisms like Remote Procedure Calls (RPC) or other web services, such as SOAP, WSDL.

Since using HTTP as a standard protocol in the architecture, REST offers all standard HTTP methods such as POST, GET, HEAD, PUT, DELETE. In addition, to develop a RESTful web service, people can use any programming languages and more importantly, there is no limitation in the platform (UNIX or Windows) while developing a RESTful web service.

To demonstrate how lightweight and simple a RESTful web service is, firstly we look at a code snippet of a SOAP-based web services.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.example.com/user">
  <pb:GetPersonalUserInfo>
   <pb:UserID>a54baf</pb:UserID>
  </pb:GetPersonalUserInfo>
 </soap:Body>
</soap:Envelope>
```

In a nutshell, the purpose of the given code is to retrieve the personal information of a user by using a userID. The code is sophisticated. Alternatively, the RESTful web service offers a much more simpler solution by sending a plain GET request to the targeting service using the following URL.

```
http://www.example.com/user/UserDetails/a54baf
```

5.2.5 Web-based Visualisation

With the flourishing of web technologies, the visualisation on the web has become trending. The visualisation on the Web extends to many topics,

including chemistry, physics and so on. Currently, there are many libraries supporting the visualisation on the web. Some examples of Javascript libraries are D3.js[3], RaphaelJS [15], Google chart [89], Processing.js [61]. In addition to Javascript language, there are other technologies that support the visualisation on the web, including Flare [2], Dynamic Data Display [3].

To build the interface for the model, we will use the Javascript library D3.js to visualise data. In this section, we will briefly explain prominent features that make D3 an excellent tool for the application.

- **Excellent documentation**
  The library has a clear and updated documentation. Besides, there are many real examples with clear instructions.
- **Active project**
  The library is frequently updated by many contributors.
- **Native language**
  Rather than using the third parties software, such as Flash (Flare) or Silverlight (Dynamic Data Display), the library is based on SVG to create the canvas. Since SVG is native to all browsers, D3 ensures the visualisation is cross-browser.
- **Flexible tool**
  D3 library is not limited in any predefined visualisation styles. Instead, it is truly a tool that allows the user to create any types of visualisation.

5.3 Use Cases

In this section, we introduce the use cases of our application. In Figure 5.2, we can see that the unique use case for the user is to *search* and for which, the use can have extended use cases as follows.

- *View file details* The use can examine the metadata of a document by hovering the mouse. A context modal containing file information will be displayed. All file information, such as the metadata, are included in the context modal.
- *Filter files by category* The clustering process will organise documents into different categories. The user can filter the search results by clicking the category name.

[2] flare
[3] https://dynamicdatadisplay.codeplex.com

Library	Advantages	Disadvantages
Google Chart	1. The library contains built-in chart templates that are ready to use 2. The developers are allowed to create custom charts	1. Although it is possible to create custom charts, it is complex to do when compare it to other libraries, for example, D3.js
RaphaelJS	1. The library uses SVG to create graphics 2. The API is easy to understand	1. The API is suitable for starters to create charts, however, there are only a small sets of functions within the API. If the developers want to create complicated charts, they have to use additional libraries 2. There are not many contributors for the project. The latest release is from August 2013
D3.js	1. The library uses SVG to construct the graphics like RaphaelsJS 2. It have a good documentation with real examples 3. There is a large community for the library 4. Many previous examples prove that it is possible to create customise charts, even the complicated charts	1. The library is big
Processing.js	1. It uses the canvas to create the graphics and animations rather than SVG	1. It is not suitable to create charts

Table 5.1: The comparison of different Javascript libraries for visualisation

- *Change result display* There are two kinds of visualisations, the schema-ball and the ordinary list. This use case demonstrates the ability of switching between two visualisations.

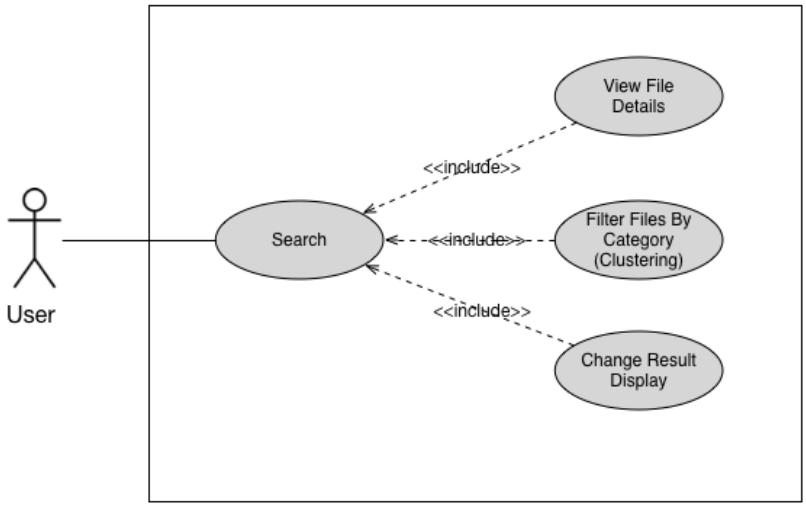


Figure 5.2: The use case diagram of the application

5.4 Prototype

In this section, we present a prototype that we have implemented for our proposed model. Figure 5.3 is a screenshot of the prototype, where we performed a search to find documents that contain either the keyword "*food*" or the keyword "*dog*" and have created date after 01-01-2013. In the figure, the top is the query builder and the bottom is the result panel where the search results are visualised. In the result panel, the user have a list of cluster names on the right panel, and the visualisation style on the left panel.

5.4.1 Query Builder

Figure 5.4 is a snapshot of the search form where we can create complex queries. The user can use many rules by clicking "Add rule", and for each rule, the user must specific the field type. At the moment, we support common fields, such as *content, file name, date created, author*. We can also extend the number of supporting fields by extending the query parser. In the figure, we present a scenario where we search for documents whose the content have

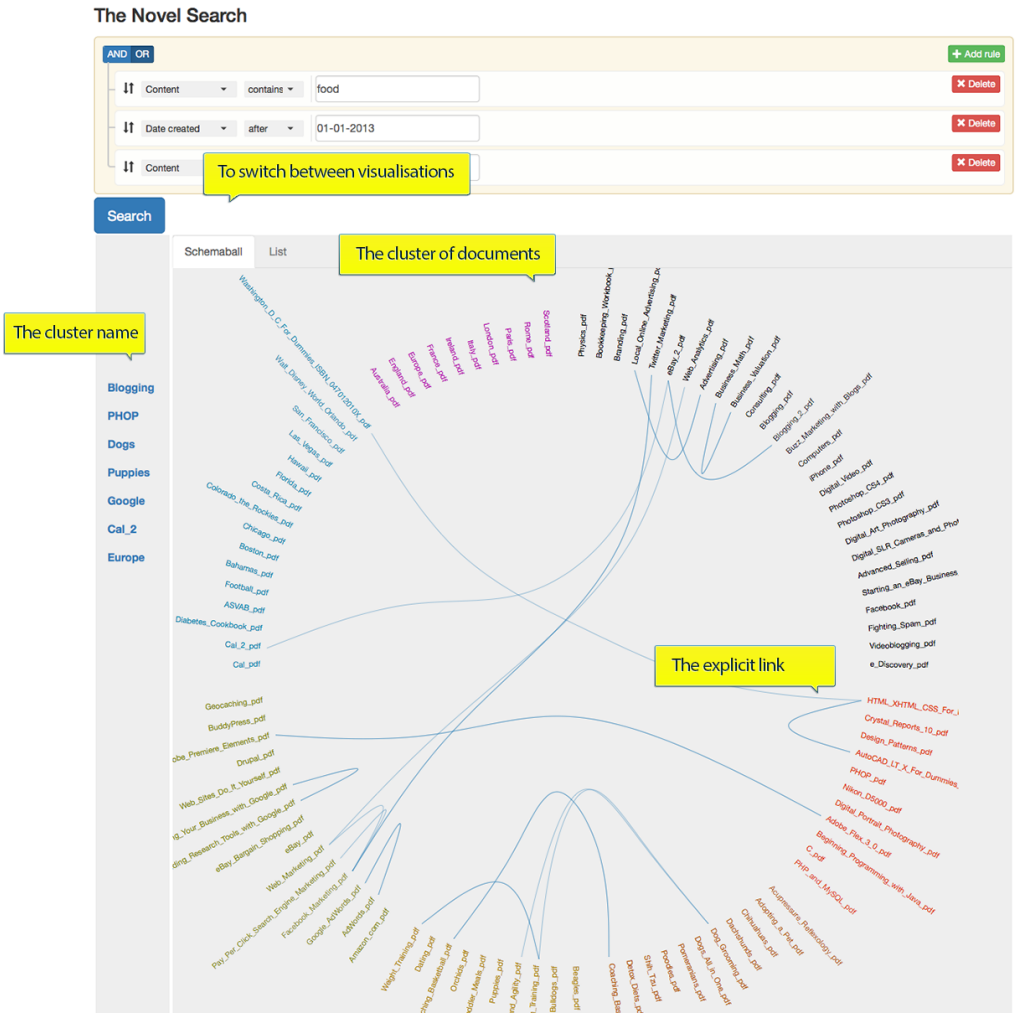


Figure 5.3: A screenshot of the prototype

two keywords "keyword1" and "keyword2", the date created after a specific date "keyword3", the author contains "keyword4" and the filename contains "keyword5".

5.4.2 Search Result Visualisation

Currently we have implemented two different visualisations. The first visualisation is the schemaball, where we can minimise the space as well as demonstrate the explicit links and the document groups on a single plane (Figure 5.5). In the schemaball, we applied the proximity principle [40] to group documents in a same category. The explicit links are visualised using the

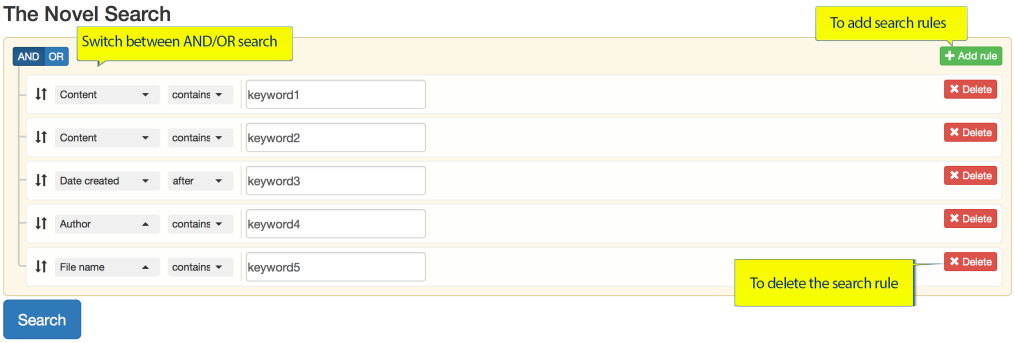


Figure 5.4: The query builder form

connecting curve. The user can view the metadata of a document by hovering (Figure 5.6) and filter documents by the category name when they click on the category name on the left panel (Figure 5.7).

The second visualisation is the ordinary list. The list consumes more space than the schemaball but in return, it displays a rich set of information for each document result, which includes the filename, the text snippet with highlighted keywords, the document path and the explicit links to other documents. Note that the list can only display documents in one cluster at a time.

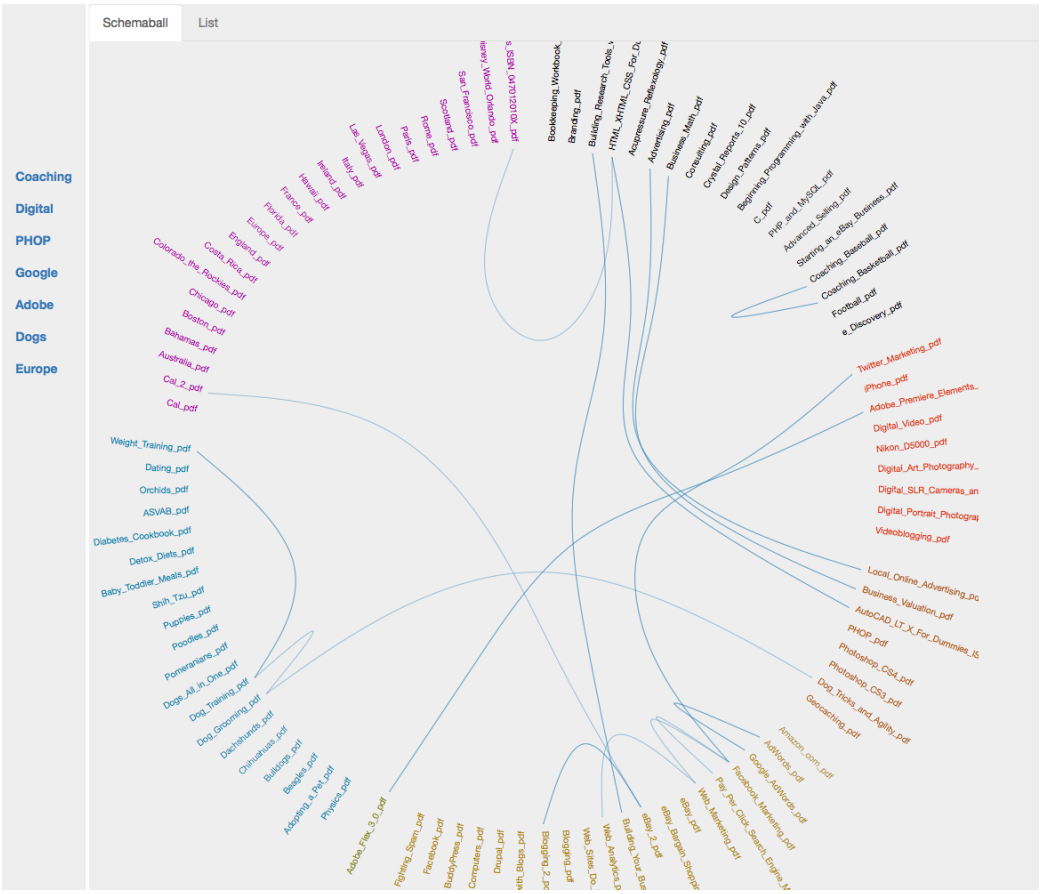


Figure 5.5: An overview of the search result using schemaball

Figure 5.6: The panel containing the information of a document displays when we hover the mouse on the document name

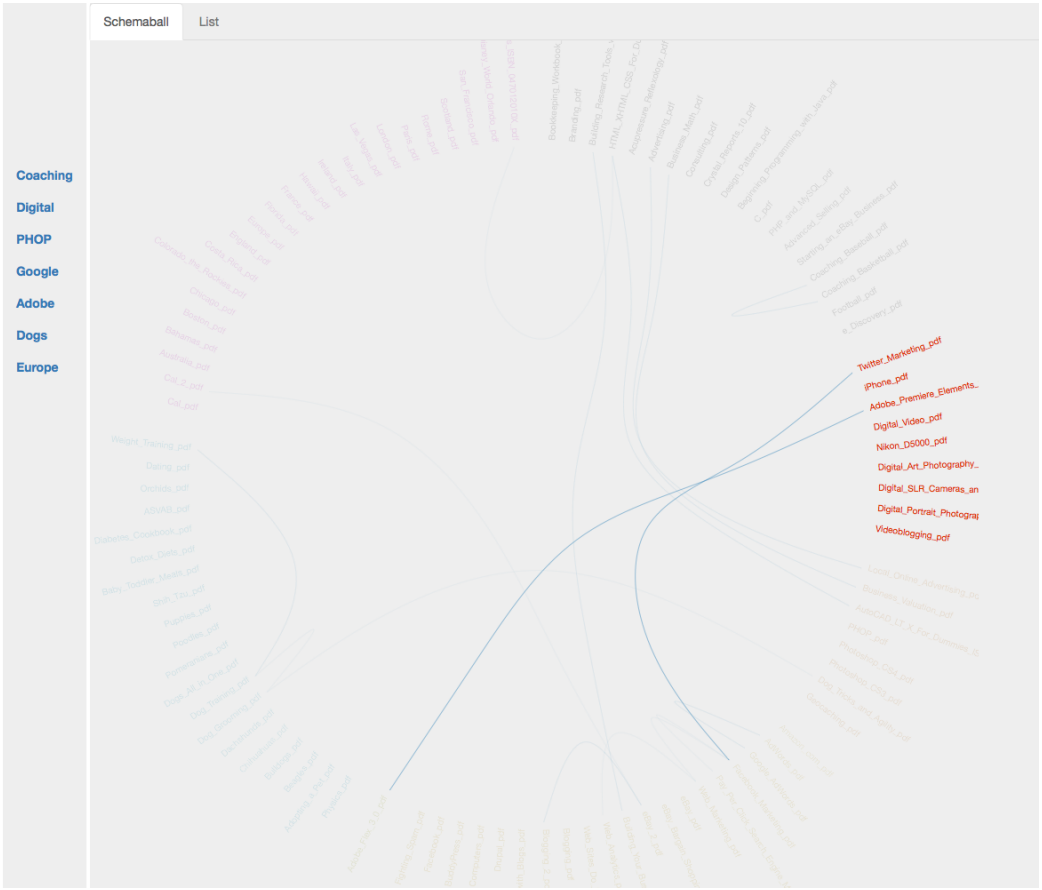


Figure 5.7: Filtering the search result using the category name on the left panel. In this case we highlighted documents of the category “digital”

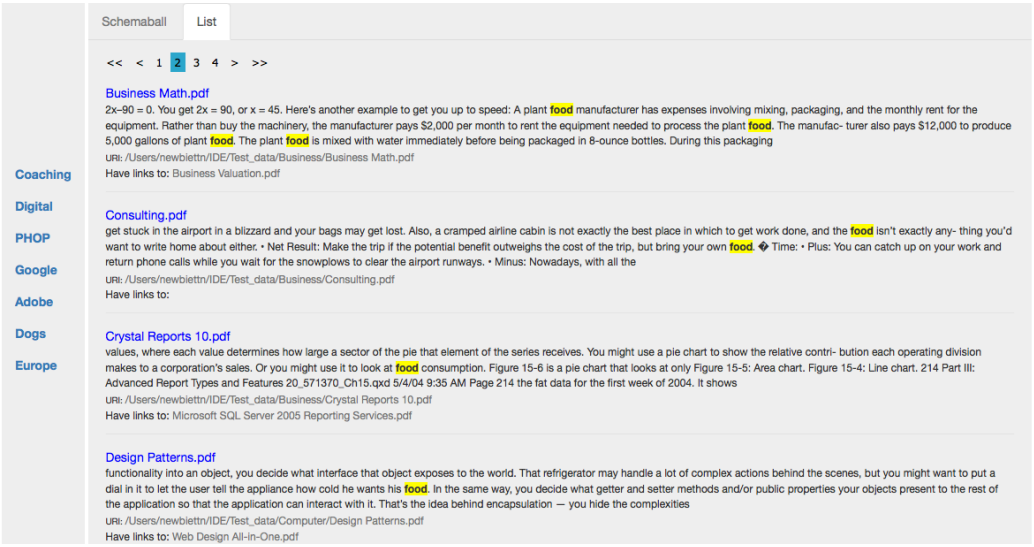


Figure 5.8: An overview of a search result using the ordinary list

6 Evaluation

6.1 Evaluation Methodology

In the evaluation, we will use both quantitative and qualitative methods to evaluate our proposed solution. According to [80], the benefit of the quantitative method is the objective and reliable results, whereas the qualitative method produces subjective, rich and detailed data. The combination of both approaches, therefore, is to compensate the weaknesses of each other. In Figure 6.1, we illustrate the model, where both quantitative and qualitative methods are used equally to produce the results. In this case, we will use the results from each method to cross-validate the evaluation outcomes.

Figure 6.1: Both quantitative and qualitative methods are integrated to gather feedback

6.2 Evaluation Goals

The ultimate goals of the user evaluation are as follows.

- Measure the usability of the proposed solution in terms of the usefulness, the ease of use, the ease of learning and the satisfaction.
- Identify the usability issues in the user interface.
- Gather user feedback for future improvements.

6.3 Evaluation Setup

The evaluation is conducted with 10 participants (5 female). They are either Master or PhD students. There are 5 participants using Windows and 5 others using Mac OS X in their daily work.

Since the focus of the usability testing is to find to which extent the application supports the user in retrieval tasks, it is unnecessary to use the personal document collections of the user. Instead, we prepared a wide range of document collections beforehand. We decided to choose the Dummies[1] how-to books as the testing data for particular advantages it offers.

- It includes a great extent of topics from accounting, photography, computing to blogging.
- The content of the book collection is not too abstract to prevent the participants from discovering.

The evaluation progress is organised as follows.

- Step 1
  We briefly explain to the user the objective of the research and the concept of the cross-document link service in terms of objective and usage (for example, what the cross-document linking is and how the link service enables the user to link between different document formats).
- Step 2
  We introduce the book collection to the user and give them five to ten minutes to explore the data. During this step, the user can use the file browsers (Windows Explorer or Mac OS Finder) and the search functionalities (Windows Search or the search box of Mac OS Finder) of the operating system to discover the book collection.

[1] http://www.dummies.com

- Step 3
  After the user feel familiar with the dataset, we explain to them our prototype:
  - How to make the boolean query by combining different fields including *content*, *title*, *author*, *date created*, *date modified*?
  - How to switch between different visualisations?
  - How to examine the schemaball visualisation to discover and browse documents?
  - How to filter the search results using the category names?
  - How to find relevant documents using the explicit links and the clusters?
- Step 4
  The user are asked to complete the questionnaire and the interview. The entire questionnaire and interview questions are included in Appendix A.

6.4 Quantitative Evaluation

We prepared 25 questions in five different parts for the quantitative evaluation. In the list below, we briefly introduce the objective of each part.

- **About you (3 questions)**: To collect the general information about the user.
- **Usefulness (6 questions)** To collect data on the usefulness of the overall application as well as specific features, including the schemaball visualisation, the explicit links, the clustering, the complex query and the text snippet.
- **Ease of Use (6 questions)** To collection information about to which degree the user can use the system effectively to accomplish their search goals.
- **Ease of Learning (3 questions)** To collect information about to which degree the user can quickly become skillful with the system.
- **Satisfaction (7 questions)** To collect information about to which degree the user are satisfied with the system.

6.5 Qualitative Evaluation

The qualitative evaluation is the semi-structured interview, where we prepare 6 main questions (Appendix A). The goal of the qualitative evaluation is to help us collect the subjective information about the system from the user's perspectives.

6.6 Results

In this section, we discuss the evaluation findings collected from the evaluation. During the discussion, we introduce both the quantitative and qualitative data to gain a better insights of the user experience. We will begin with the general information and go deep into specialised features including the explicit links, the clustering, the visualisation and the enhanced search form.

6.6.1 General

In general, we received promising feedback from the participants about the application in both the quantitative data (Figure 6.2) as well as the qualitative data. Firstly, we asked the participants about the advantages of the application. All participants answered they were delighted at seeing the relationships between documents in terms of the explicit and implicit links. Particularly, there were 3 participants clearly indicated because they are research students, they have to work with many journals in different topics. Normally, most of the journals are on the desktop and they have ambiguous file names, therefore, realising relationships between documents is valuable to their work. "*It is really a great application because I can find the relationships between documents, especially when you have to do the Master thesis. You have many references and you want to find the relationships between documents. You do not have any application does that*", one said. Secondly, 8 participants showed their interests in using the content-based search. It is an interesting finding because both Windows and Mac OS X support this feature, but there were three Windows participants had no knowledge about this availability beforehand. "*I have not searched for the content like this before. I can only search with the title in Windows*", a Windows participant said. Also, all 5 participants who use Mac OS are not satisfied with the content-based search feature of Mac. They complained that the system gives no clue about the document content.

Apart from the mentioned advantages, all participants reported the slow

performance of the application. However, one participant said, "*In case I want to find the relationships between documents, I can accept the lagging time*". More critically, when we asked a participant about the problems she can encounter when she uses Windows Explorer to manage her documents, she replied she usually does not have many problems. "*I only have a small number of folders. I know where a specific document is*", she explained how she retrieves a document. "*The application is more applicable to the library where you have many documents*", she commented. Finally, one interesting idea we recorded during the interview is one participant suggested that she would prefer to use the application as a plugin of Windows Explorer or Mac OS Finder, rather than an independent application.

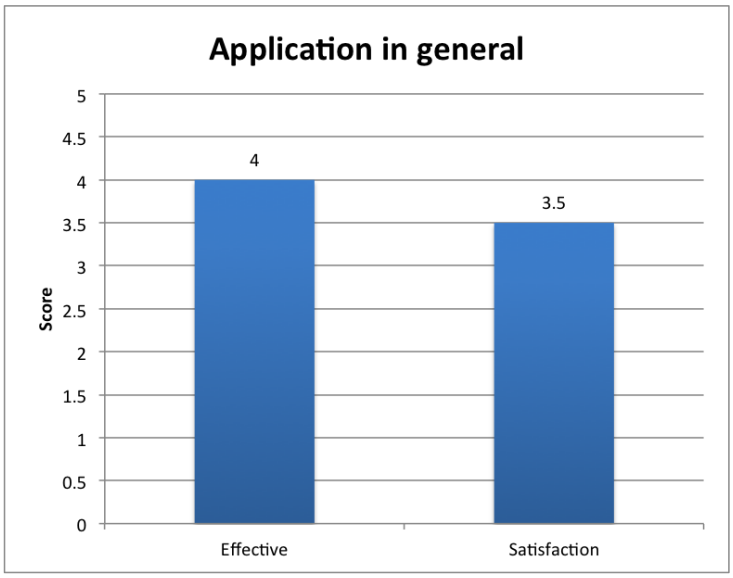


Figure 6.2: The opinions about the application in general

6.6.2 Explicit Links

Despite the fact that all participants had no knowledge about the concept of explicit links in advance, the collected data from both quantitative and qualitative methods confirm the contribution of the explicit links to document retrieval on the desktop. Figure 6.3 shows the mean scores of the explicit links in terms of the usefulness, the ease of learning and the satisfaction. Whereas the scores of the usefulness and the satisfaction indicate the explicit links are very much beneficial to all participants, the score of the ease of learning gives a hint about the unfamiliarity of the concept to them.

During the interview, 9 of 10 participants gave positive comments on the explicit links. One said, "*Maybe it is good for doing something like literature search or something like that*". Another participant said, "*It is like a recommendation to another document*". Nevertheless, because we had created the explicit links ourselves early instead of letting the participants create the explicit links themselves, one participant expressed her trouble when we asked if she could easily find the relevant documents via the explicit links. "*I did not read the content of documents so I cannot find if it is useful*", she stated. One also expressed his frustration with being unfamiliar with the dataset, so he could not confirm if the explicit links suggested relevant documents. This drawback can partly explain the lower score of the ease of use when compare it to the usefulness and satisfaction. In contrast, one particular did not like the explicit links. "*If you have too many links, when you visualise, it will not make sense. It will be like a web*", she said.

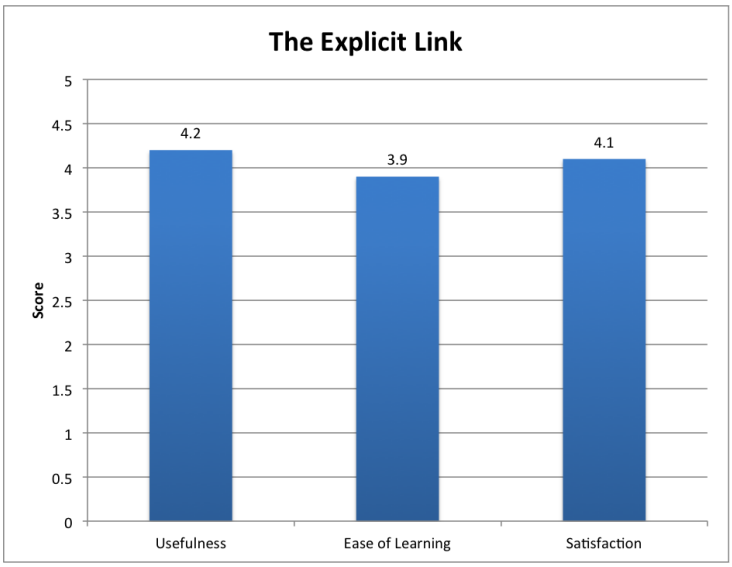


Figure 6.3: The opinions about the explicit links

6.6.3 Clustering

Figure 6.4 shows the mean score of the clustering feature in terms of the usefulness, the ease of use and the satisfaction. The result is very encouraging, indicating the need of an automatic support to the retrieval tasks of the user. During the interview, one participant clearly mentioned the document clustering is an advantage. "*I can see which documents are linked together*, she supported her claim. Another participant said the clusters are really

useful because she can see the topics of different groups and can differentiate them through the colors of the clusters. In contrast, one participant was not happy about the name of the clusters. "*I am afraid that sometimes the cluster name does not relate to the content of documents*", he said. Moreover, in the same manner as the explicit links, being unfamiliar with the dataset made the participants frustrated when we asked them to evaluate the correctness of the clustering.

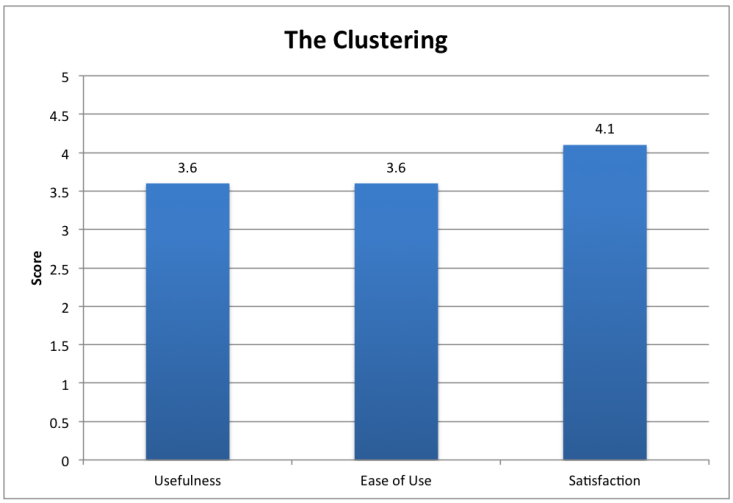


Figure 6.4: The opinions about the clustering

6.6.4 Visualisation

The participants expressed their interests towards the visualisation. They were interested in seeing different topics in different colors, the visualisation of explicit links as well as multiple visualisations. One stated that using different colors for different clusters are intuitive, so she can know to which category documents belong. Another comment on the interface is when she tried to search by multiple keywords, instead of a single text snippet, she suggested that our application should have multiple text snippets where each of them containing a corresponding keyword. One participant also gave a hint about the needs of arrows for the explicit links to indicate the direction of the link. For the inquiry regarding to additional visualisations for the system, 9 of 10 participants thought that they were satisfied with two current visualisations, whereas one participant suggested another kind of visualisation that supports hierarchical clustering where he can zoom in and out by the topic name. "*For example, in the information technology topic, you put*

subtopics such as machine learning, computing", he described his idea about the visualisation. About the multiple visualisations, he gave us a positive comment, "*We prefer to have more options, because it is easier to represent*".

6.6.5 Enhanced Search Form

The user user evaluation reveals an interesting fact that while all Mac OS participants know the search system supports content-based search, 4 out of 5 Windows participants did not know this features is also supported in Windows. The boolean query was interesting to all of participants, indicating the needs of a more flexible search in both Windows and Mac OS X.

6.7 Conclusion

Both the collected quantitative and qualitative data state our proposed solution consisting the explicit links, the clustering, the advanced search features is contributive to the document retrieval. Valuable suggestions from the participants open the model to new possibilities including the hierarchical clustering, the multiple text snippets and the additional search fields. Finally, it is critical to improve the search time in the future.

7 Discussion and Future Work

7.1 Discussion

The ultimate goal of this thesis was to enhance retrieval and discovery of desktop documents by focusing further on the idea visioned by Vannevar Bush that document retrieval should be analogous to the human brain, where people are capable of retrieving them effortlessly using a given hint [6]. To accomplish the goal, we have proposed a novel model where we take into account both explicit and implicit links between documents. Whereas the novel model defines the implicit link using text data mining techniques for document's metadata and content, the explicit link, which is analogous to the associative trails of the memex, is enabled using the link service [84]. In this thesis, we have outlined a number of possibilities that explicit links can contribute to the desktop retrieval and within the scope of this thesis, we have utilised the use case that explicit links can be used to suggest relevant documents.

Besides the explicit and implicit links, the novel model that we have proposed is adapted to concur specific requirements. First, we have emphasised the issue of the multitude of digital document formats and designed an extensible parser that is capable of resolving the issue. Furthermore, we enhance the search features by means of boolean queries and wildcards, and designed an extensible architecture that can adopt different document formats as well

as visualisation styles. Finally, we have instantiated the model by an implemented interface with two different visualisations, the schemaball and the ordinary list. To evaluate the solution, we conducted an evaluation with end users using both quantitative and qualitative methods. The result of the user study is optimistic, indicating the model would be useful in supporting the retrieval task on the desktop.

Our proposed model has overcame many of the shortcomings of previous researches. First, none of those considers the content of documents. They only focus on the metadata to organise the document collection. Furthermore, they do not consider the explicit links to enhance the retrieval tasks and support the retrieval tasks with enhanced search features, such as boolean queries or wildcards. Finally, their visualisations are limited because they only focus on a specific visualisation.

7.1.1 Contributions

To summarise, we outline the contributions of our research as follows.

1. We identified challenges of document retrieval and discovery on the desktop by making a comprehensive review. The review includes both built-in search systems and different techniques that have been proposed.

2. Based on the results of our review, we have proposed a novel model for document retrieval and discovery on the desktop. It first takes full advantage of the document attributes as well as the content to support the retrieval task. By exploiting the explicit links between documents, it ensures the search results are more relevant and personalised to the user.

3. We designed an extensible architecture that can adopt different digital document formats as well as visualisation styles.

4. We implemented a prototype as a proof of concept for the proposed model. To illustrate, we implemented two different visualisation styles for the prototype, the schemaball and the ordinary list.

5. To evaluate the solution, we conducted an evaluation with end-users. The results of our evaluation confirm the potential of our proposed model in supporting retrieval and discovery of documents on the desktop.

7.2 Future Work

7.2.1 Exploit Further Possibilities of Explicit Links

In this thesis, we have outlined a number of use cases where the explicit links can contribute to the document retrieval task. So far we have only taken into consideration the use case that the explicit link can suggest relevant documents to the user. We believe that there is still room for future researches with remaining use cases: (i) use explicit links to improve clustering quality, (ii) mining explicit links to suggest further links to the user, (iii) study the explicit links at the paragraph level (i.e. take into consideration the exact location of the explicit link within the document).

7.2.2 Investigate the Performance of Synonym-based Search

During the implementation, we realised that the idea of using WordNet to support synonym-based drastically decreases the search speed of the system because we have to make multiple requests for a single search as well as remote query to WordNet. In future research, if we can address the performance issue of the synonym-based search then we can open new possibilities to document retrieval where we can retrieve more relevant documents.

A Appendix

Quantitative Questions

About You

1. **What is your gender?**
   - ☐ Male
   - ☐ Female
2. **What is your academic position?**
   - ☐ Bachelor
   - ☐ Master
   - ☐ Phd
   - ☐ Other _____
3. **What operating system (OS) do you use?**
   - ☐ Windows
   - ☐ Mac OS
   - ☐ Other

Usefulness

4. **To what degree do you think the Schemaball visualisation is useful in organising documents (1 = not at all, 5 = extremely useful) ?**
   - ☐ 1
   - ☐ 2
   - ☐ 3
   - ☐ 4
   - ☐ 5
5. **To what degree do you think the explicit link between documents is useful in helping you find and discover documents (1 = not at all, 5 = extremely useful) ?**
   - ☐ 1
   - ☐ 2
   - ☐ 3
   - ☐ 4
   - ☐ 5
6. **To what degree do you think the automatic categorisation (the keyword on the left) is useful in helping you find and discover documents according to the topic (1 = not at all, 5 = extremely useful) ?**
   - ☐ 1
   - ☐ 2
   - ☐ 3
   - ☐ 4
   - ☐ 5
7. **To what degree do you think the search form of our application is useful in giving you more control over the search query (1 = not at all, 5 = extremely useful) ?**
   - ☐ 1
   - ☐ 2
   - ☐ 3
   - ☐ 4
   - ☐ 5

8. **To what degree do you think the text snippet in the search results is useful in finding and discover documents (1 = not at all, 5 = extremely useful) ?**
   - ☐ 1
   - ☐ 2
   - ☐ 3
   - ☐ 4
   - ☐ 5
9. **To what degree do you think our application is effective in finding and discovering documents (1 = not at all, 5 = extremely effective) ?**
   - ☐ 1
   - ☐ 2
   - ☐ 3
   - ☐ 4
   - ☐ 5

Ease of Use

10. **To what degree do you think our application visualisation is easy to use (1 = not at all, 5 = extremely easy) ?**
    - ☐ 1
    - ☐ 2
    - ☐ 3
    - ☐ 4
    - ☐ 5
11. **To what degree do you think using the query model (AND, OR, *) is easy to use (1 = not at all, 5 = extremely easy) ?**
    - ☐ 1
    - ☐ 2
    - ☐ 3
    - ☐ 4
    - ☐ 5

12. To what degree do you think you can use the application without instructions (1 = totally impossible, 5 = totally possible) ?

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

13. To what degree do you think the automatic categorisation of documents is consistent when you use it (1 = not consistent at all, 5 = extremely consistent) ?

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

14. To what degree do you think changing between different visualisation is easy to use (1 = not consistent at all, 5 = extremely easy) ?

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

15. To what degree do you think you can finding detail information about a document by hovering the mouse on it is easy to use (1 = not consistent at all, 5 = extremely easy) ?

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

Ease of Learning

16. **To what degree do you think you can learn to use the search form quickly (1 = very difficult to learn, 5 = extremely easy to learn) ?**
    - ☐ 1
    - ☐ 2
    - ☐ 3
    - ☐ 4
    - ☐ 5
17. **To what degree do you think you can learn to use the explicit links quickly to find relevant documents (1 = very difficult to learn, 5 = extremely easy to learn) ?**
    - ☐ 1
    - ☐ 2
    - ☐ 3
    - ☐ 4
    - ☐ 5
18. **To what degree do you think you can quickly became skilful with our application (1 = very difficult to learn, 5 = extremely easy to learn) ?**
    - ☐ 1
    - ☐ 2
    - ☐ 3
    - ☐ 4
    - ☐ 5

Satisfaction

19. **To what degree do you think you are satisfied with the system in general (1 = not satisfied at all, 5 = extremely satisfied) ?**
    - ☐ 1
    - ☐ 2
    - ☐ 3
    - ☐ 4

□ 5

20. To what degree do you think you are satisfied with using the search form with different fields to search for documents (1 = not satisfied at all, 5 = extremely satisfied) ?

□ 1

□ 2

□ 3

□ 4

□ 5

21. To what degree do you think the the search form with different fields is pleasant to use (1 = not pleasant at all, 5 = extremely pleasant) ?

□ 1

□ 2

□ 3

□ 4

□ 5

22. To what degree do you think you are satisfied with the documents you find and discover through explicit links (1 = not satisfied at all, 5 = extremely satisfied) ?

□ 1

□ 2

□ 3

□ 4

□ 5

23. To what degree do you think you are satisfied with the documents you find and discover when documents in the same topic are grouped together (1 = not satisfied at all, 5 = extremely satisfied) ?

□ 1

□ 2

□ 3

□ 4

□ 5

24. **To what degree do you think you are satisfied with the documents you find and discover when using our application(1 = not satisfied at all, 5 = extremely satisfied) ?**
    - □ 1
    - □ 2
    - □ 3
    - □ 4
    - □ 5
25. **To what degree do you think you are satisfied with the documents you find and discover when using built-in search systems in Windows or Mac OS (1 = not satisfied at all, 5 = extremely satisfied) ?**
    - □ 1
    - □ 2
    - □ 3
    - □ 4
    - □ 5

Qualitative Questions

1. In your opinion, what are the advantages of our application?
2. In your opinion, what are the disadvantages of our application?
3. In your opinion, what suggestions do you have for improvement?
4. In your opinion, what are the differences between our system and the system of Mac/Windows?
5. In your opinion, do you think we need more visualisations?
6. In your opinion, what do you think about the performance of our application? Is it slow or fast?

Bibliography

[1] A. R. Barton, V. L. Schatz, and L. N. Caplan. Information Retrieval on a High-Speed Computer. In *Proceedings of the Western Joint Computer Conference*, pages 77–80, 1959.

[2] S. Björk. Hierarchical Flip Zooming: Enabling Parallel Exploration of Hierarchical Visualizations. In *Proceedings of AVI 2000*, pages 232–237, 2000.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D3 Data-Driven Documents. *IEEE TVCG*, 17:2301–2309, 2011.

[4] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

[5] K. Burnett, K. B. Ng, and S. Park. A Comparison of the Two Traditions of Metadata Development. *Journal of the American Society for Information Science*, 50:1209–1217, 1999.

[6] V. Bush. As We May Think. *The Atlantic Monthly*, 176(1):101–108, 1945.

[7] Y. Cai, X. L. Dong, A. Halevy, J. M. Liu, and J. Madhavan. Personal Information Management with SEMEX. In *Proceedings of SIGMOD '05*, pages 921–923, 2005.

[8] S. K. Card, J. D. Mackinlay, and B. Shneiderman. The FISHEYE View: A New Look at Structured Files. In S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors, *Readings in Information Visualization*, pages 312–330. Morgan Kaufmann Publishers, 1998.

[9] P. R. Christopher Manning and H. Schutze. *Introduction to Information Retrieval*, chapter 1, pages 1–17. Cambridge University Press, 2008.

[10] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM CSUR*, 41:1–31, 2008.

[11] A. Cockburn and B. Mckenzie. 3D or Not 3D? Evaluating the Effect of the Third Dimension in a Document Management System. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 434–441, 2001.

[12] A. Cockburn and B. Mckenzie. Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–210, 2002.

[13] D. Cutting and J. Pedersen. Optimization for Dynamic Inverted Index Maintenance. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 405–411, 1990.

[14] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[15] D. D. M. . D. Dawber. *Learning Raphaël JS Vector Graphics*. Packt Publishing, 2013.

[16] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *JASIS*, 41(6):391–407, 1990.

[17] X. Dong, A. Halevy, and J. Madhavan. Reference Reconciliation in Complex Information Spaces. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 85–96, 2005.

[18] P. Dourish, W. K. Edwards, A. Lamarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton. Extending Document Management Systems with User-Specific Active Properties. *ACM Transactions on Information Systems*, 18:140–170, 2000.

[19] S. T. Dumais and T. K. Landauer. Using Examples to Describe Categories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 112–115, 1983.

[20] E. N. Efthimiadis and A. Carlyle. Organizing Internet Resources: Metadata and the Web. *Bulletin of the American Society for Information Science and Technology*, 24:4–5, 1997.

[21] D. C. Engelbart. Augmenting Human Intellect: A Conceptual Framework. In N. Wardrip-Fruin and N. Montfort, editors, *The New Media Reader*. MIT Press, 1962.

[22] E. Freeman and D. Gelernter. Lifestreams: A Storage Model for Personal Data. *SIGMOD Record*, 25(1):80–86, 1996.

[23] E. Freeman and D. Gelernter. Beyond Lifestreams: The Inevitable Demise of the Desktop Metaphor. *Kaptelinin and Czerwinski*, pages 19–48, 2007.

[24] G. W. Furnas. Generalized Fisheye Views. *CHI '86*, 17(4):16–23, 1986.

[25] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. Mylifebits: Fulfilling the Memex Vision. In *Proceedings of the Tenth ACM International Conference on Multimedia*, pages 235–238, 2002.

[26] M. Golemati, a. Katifori, E. Giannopoulou, I. Daradimos, and C. Vassilakis. Evaluating the Significance of the Windows Explorer Visualization in Personal Information Management Browsing Tasks. In *Proceedings of IV'07*, pages 93–100, 2007.

[27] A. Griffiths, H. C. Luckhurst, and P. Willett. Using Interdocument Similarity Information in Document Retrieval Systems. *Journal of the American Society for Information Science*, 37:3–11, 1986.

[28] F. G. Halasz, T. P. Moran, and R. H. Trigg. Notecards in a Nutshell. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*, pages 45–52, 1987.

[29] H. Haller and A. Abecker. Imapping: A Zooming User Interface Approach for Personal and Semantic Knowledge Management. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, pages 119–128, 2010.

[30] E. Hatcher and O. Gospodnetic. *Lucene in Action*. Manning Publications, 2 edition, 2004.

[31] M. A. Hearst and J. O. Pedersen. Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. In *Proceedings of the 19th Annual*

International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 76–84, 1996.

[32] S. Henderson. Genre, Task, Topic and Time: Facets of Personal Digital Document Management. In *Proceedings of the 6th ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction: Making CHI Natural*, pages 75–82, 2005.

[33] A. Hotho, S. Staab, and G. Stumme. Ontologies Improve Text Document Clustering. In *The Third IEEE International Conference on Data Mining*, pages 541–544, 2003.

[34] D. A. Hull. Stemming Algorithms: A Case Study for Detailed Evaluation. *Journal of the American Society for Information Science and Technology*, 47(1):70–84, 1996.

[35] B. Johnson and B. Shneiderman. Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. In *Proceedings of the 2nd Conference on Visualization '91*, pages 284–291, 1991.

[36] S. C. Johnson. Hierarchical Clustering Schemes. *Psychometrika*, 32(3):241–254, 1967.

[37] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data. In G. W. Michael Stonebraker and D. Dewitt, editors, *Proceedings of the CIDR Conference*, pages 13–28, 2005.

[38] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. Websom "Self-Organizing Maps of Document Collections". *Neurocomputing*, 21:101–117, 1998.

[39] A. Katifori, C. Vassilakis, and A. Dix. Ontologies and the Brain: Using Spreading Activation through Ontologies to Support Personal Interaction. *Cognitive Systems Research*, 11(1):25–41, 2010.

[40] K. Koffka. The Environmental Field. In *Principles of Gestalt Psychology*, pages 164–165. Taylor & Francis, 2013.

[41] T. Kohonen. The Self-Organizing Map. *Neurocomputing*, 21:1–6, 1998.

[42] S. Koshman, A. Spink, and B. J. Jansen. Web Searching on the Vivisimo Search Engine. *Journal of the American Society for Information Science and Technology*, 57:1875–1887, 2006.

[43] R. L. Kullberg. Dynamic Timelines - Visualizing Historical Information in Three Dimensions. Master's thesis, MIT Media Lab, 1995.

[44] R. M. L. Page, S. Brin and T. Winograd. The Pagerank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[45] J. Lamping, R. Rao, and P. Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 401–408, 1995.

[46] M. W. Lansdale. The Psychology of Personal Information Management. *Applied Ergonomics*, 19:55–66, 1988.

[47] H. Luhn. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[48] Maarten, Maarten, M. P. Czerwinski, M. Van Dantzich, G. Robertson, and H. Hoffman. The Contribution of Thumbnail Image, Mouse-over Text and Spatial Location Memory to Web Page Retrieval in 3D. *Proceedings of the INTERACT'99*, pages 163–170, 1999.

[49] R. Madsen, S. Sigurdsson, L. Hansen, and J. Larsen. Pruning the Vocabulary for Better Context Recognition. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 2, pages 483–488, 2004.

[50] T. W. Malone. How Do People Organize Their desks?: Implications for the Design of Office Information Systems. *ACM Transactions on Information Systems*, 1(1):99–112, 1983.

[51] M. Maron, J. Kuhns, and L. Ray. Probabilistic Indexing. A Statistical Technique for Document Identification and Retrieval. Technical report, DTIC Document, 1959.

[52] G. a. Miller. Wordnet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[53] G. W. Milligan. An Examination of the Effect of Six Types of Error Perturbation on Fifteen Clustering Algorithms. *Psychometrika*, 45(3):325–342, 1980.

[54] G. Mosweunyane, L. Carr, and N. Gibbins. A Tag-like, Linked Navigation Approach for Retrieval and Discovery of Desktop Documents. In H. Cherifi, J. Zain, and E. El-Qawasmeh, editors, *Digital Information and Communication Technology and Its Applications*, Communications in Computer and Information Science, pages 692–706. Springer Berlin Heidelberg, 2011.

[55] T. H. Nelson. Complex Information Processing: A File Structure for the Complex, the Changing and the Indeterminate. In *Proceedings of the 1965 20th National Conference*, pages 84–100, 1965.

[56] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 40(3):211–218, 2006.

[57] D. M. Powers. Evaluation: From precision, Recall and F-Measure to Roc, Informedness, Markedness and Correlation. *Journal of Machine Learning Research Homepage*, 2:37–63, 2011.

[58] S. E. Preece. Clustering as An Output Option. *Proceedings of the American Society for Information Science*, 10:189–190, 1973.

[59] P. Ravasio, S. G. Schär, and H. Krueger. In Pursuit of Desktop Evolution: User Problems and Practices with Modern Desktop Systems. *ACM Transactions on Computer-Human Interaction*, 11(2):156–180, 2004.

[60] P. Ravasio and V. Tscherter. The Users' Theories on the Desktop Metaphor - or Why We Should Seek Metaphor-Free Interfaces. In *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments*, 2007.

[61] J. Resig, B. Fry, and C. Reas. Processing.js, 2008.

[62] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.

[63] G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. Van Dantzich. Data mountain: Using Spatial Memory for Document Management. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, pages 153–162, 1998.

[64] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone Trees: Animated 3d Visualizations of Hierarchical Information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 189–194. ACM, 1991.

[65] S. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, 33:294–304, 1977.

[66] G. Salton. Search Strategy and the Optimization of Retrieval Effectiveness. *Proceedings of the FID/IFIP Conference on Mechanized Information Storage, Retrieval, and Dissemination*, 1967.

[67] G. Salton. Automatic Information Organization and Retrieval. *Computer*, pages 41–56, 1980.

[68] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[69] G. Salton, a. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[70] G. Salton and C. Yang. On the Specification of Term Values in Automatic Indexing. *Journal of Documentation*, 29:351–372, 1973.

[71] M. L. Scott. *Dewey Decimal Classification*, chapter General Aspects of the Dewey Decimal Classification, pages 13–19. Libraries Unlimited, 21st edition, 2005.

[72] J. Sedding and D. Kazakov. Wordnet-Based Text Document Clustering. In *Proceedings of the 3rd Workshop on Robust Methods in Analysis of Natural Language Data*, pages 104–113, 2004.

[73] S. Sharma and Vishal Gupta. Recent Developments in Text Clustering Techniques. *International Journal of Computer Applications*, 37(6):14–19, 2012.

[74] D. Shenk. *Data Smog: Surviving the Information Glut*, chapter The Laws of Data Smog. Harper San Francisco, 2009.

[75] B. Signer and M. Norrie. As We May Link: A General Metamodel for Hypermedia Systems. In *ER 2007*, volume 4801, pages 359–374, 2007.

[76] R. R. Sokal. Numerical Taxonomy. *Scientific American*, 215(6):106–116, 1966.

[77] K. Sparck Jones. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[78] J. Stasko. An Evaluation of Space-filling Information Visualizations for Depicting Hierarchical Structures. *International Journal of Human-Computer Studies*, 53(5):663–694, 2000.

[79] J. Stasko and E. Zhang. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. In *IEEE Symposium on Information Visualization 2000*, pages 57–65. IEEE, 2000.

[80] A. Steckler, K. R. Mcleroy, R. M. Goodman, S. T. Bird, and L. Mccormick. Toward Integrating Qualitative and Quantitative Methods: An Introduction. *Health Education Quarterly*, 19:1–8, 1992.

[81] M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. In *In KDD Workshop on Text Mining*, volume 400, pages 525–526, 2000.

[82] P. Switzer. Vector Images in Document Retrieval. *Statistical Association Methods for Mechanized Documentation*, pages 163–171, 1965.

[83] M. Taube, C. D. Gull, and I. S. Wachtel. Unit Terms in Coordinate Indexing. *American Documentation*, 3:213–218, 1952.

[84] A. Tayeh and B. Signer. Open Cross-Document Linking and Browsing Based on a Visual Plug-in Architecture. *Proceedings of WISE 2014*, pages 231–245, 2014.

[85] C. P. Thacker, E. Maccreight, and B. W. Lampson. Alto: A Personal Computer. In B. Siewiorek and Newell, editors, *Computer Structures: Principles and Examples*, pages 549–572. Xerox, Palo Alto Research Center, 2nd edition, 1979.

[86] A. Tombros, A. Tombros, R. Villa, and C. J. Van Rijsbergen. The Effectiveness of Query-Specific Hierarchic Clustering. *Information Processing and Management*, 38:559–582, 2002.

[87] O. Zamir and O. Etzioni. Grouper: A Dynamic Clustering Interface to Web Search Results. *Computer Networks*, 31(11-16):1361–1374, 1999.

[88] S. Zhong and J. Ghosh. Generative Model-Based Document Clustering: A Comparative Study. *Knowledge and Information Systems*, 8(3):374–384, 2005.

[89] Y. Zhu. Introducing Google Chart Tools and Google Maps API in Data Visualization Courses. *IEEE Computer Graphics and Applications*, 32(6):6—9, 2012.