Hamburg University of Technology

Accessibility Assistance for the Interactive Navigation of Texts

Author:
Imad HAMOUMI

Supervisors:
Dr. Patricio FARRELL
Dr. Julian KUNKEL

MASTER THESIS

Institute of Mathematics

January 28, 2019

Declaration of Authorship

I, Imad HAMOUMI, declare that this thesis titled, "Accessibility Assistance for the Interactive Navigation of Texts" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

HAMBURG UNIVERSITY OF TECHNOLOGY

Abstract

Institute of Mathematics

Master Thesis

Accessibility Assistance for the Interactive Navigation of Texts

by Imad HAMOUMI

A mass increase of information has been observed in the last years. The web has undergone a phase of rapid growth regarding content and became an important medium. This growth led to new challenges for users to satisfy their information needs, as well as for service providers to store content and make it easily accessible. Researches have proposed new approaches that improve the quality of search machines, the accuracy of ranking algorithms, and the performance of storing systems. However, they neglect the role of the interface that visually presents this information and assists the user in finding it. Thus, these approaches are only usable if the interface used is optimised to serve the users.

This thesis describes the dominant approach used today to search and explore text data. Then, it proposes a navigation model that improves these approaches. In order to proof the theory of this navigation model, three interfaces are implemented. These interfaces integrate the data of a service that is used to search for research data. This service serves as case study to evaluate the results. The evaluation is conducted with test users and discussed at the end of the thesis.

Acknowledgements

The author would like to thank his supervisors, Dr. Patricio FARRELL and Dr. Julian KUNKEL, for valuable ideas, support and motivation.

Contents

List of Figures

List of Tables

List of Abbreviations

EUDAT	**E**uropean **D**ata **I**nfrastructure
GUI	**G**raphical **U**ser **I**nterface
UI	**U**ser **E**rxperience
SAS	**S**ystem **U**sability **S**cale
DKRZ	**D**eutsches **K**limarechenzentrum
API	**A**pplication **P**rogramming **I**nterfaces
JSON	**J**avaScript **O**bject **N**otation
OAI-PMH	**O**pen **A**rchives **I**nitiative **P**rotocol for **M**etadata **H**arvasting
CKAN	**C**omprehensive **K**nowledge **A**rchive **N**etwork
CESSDA	**C**onsortium of **E**uropean **S**ocial **S**cience **D**ata **A**rchives
UH	**U**niversität **H**amburg
TUHH	**H**amburg **U**niversity of **T**echnology
VSM	**V**ector **S**pace **M**odel
BOW	**B**ag- **o**f- **W**ords
DF	**D**ocument **F**requency
TF	**T**erm **F**requency
IDF	**I**nversed **D**ocument **F**requency
SVM	**S**upport **V**ector **M**achines
NB	**N**aive **B**ayes
TP	**T**rue **P**ositive
TN	**T**rue **N**egative
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
HIV	**H**uman **I**mmunodeficiency **V**irus

Chapter 1

Introduction

"When you apply computer science and machine learning to areas that haven't had any innovation in 50 years, you can make rapid advances that seem really incredible." - Bill Maris

1.1 Motivation

With the rapid development of digital technologies and smart devices, a large amount of data is being generated every day. This data has become more relevant and a major priority for businesses companies, who started collecting and storing it [23, 13]. As a result, various techniques have been proposed, with the aim to manage such amount of data and make it useful. But what does *useful* actually mean?. According to [25, 10], the larger part of data collected by companies today consists of structured and unstructured textual data. Companies analyse and explore textual data to gain new insights about their customers and to understand their surfing behaviour, to improve business processes, or to predict future developments.

One of the important tasks when possessing data is to make it easily accessible. Consider, for example, a news company website with millions of daily visits, who produces gigabytes of news articles every week. This company can process this data to generate business values based on political decisions such as predicting stock market crashes, forecasting product prices, or even providing supports to election campaigns. However, possessing this articles is the positive aspect, the other aspect is how this huge amount of articles can be accessed. The result is that the visitors are exposed to much more information than they used to be.

In the last years, information retrieval (IR) techniques assisted to extract important informations from huge amount of sources. These techniques focused on approaches that can be used to retrieve relevant information, and avoid overwhelming the users [60]. As a result, new tools called search engines were born. These have emerged as efficient techniques for finding relevant information. They are built focusing on query streams, and optimized based on factors such as response time, storage size, ranking and quality of the search results. However, their designers ignored to ask *why* the users are performing their searches, and *how* the result can be presented [44].

The question *why* a user is posing a query is actually essential to satisfy his/her need. The intention behind a search process is not random. No user opens the browser and says "I think I will do some searches" [44]. A search request has always an objective and is merely a way to satisfy an underlying goal that has to be

achieved. This goal may be looking for the next car to buy, information about an operating system, or looking for a pet to adopt. In fact, all the goals of these examples can be conveyed by the same query, namely **"Buy a Jaguar"**. The result would be a mixed list of documents containing content about the Jaguar car brand, the Jaguar operating system by Apple, and about the animal jaguar.

Many researchers assume that only a high performing search engine improves the search behavior. However, the search representation is also important. The result of the search query is just back-bone of the search system, the interface is responsible for presenting the results to the user. If this result is presented in lists that are too long, the user could just visit the first documents. If the result is not well categorised, the user could leave the system. Generally, improving the user interface of the search engine will optimise the result representation, and the user will automatically navigate to the specific information faster, which would yield to more insight about the results [3].

While much work has been done on efficient processing of huge volumes of text data [23], little work considers how to present them, and how to make them easy navigable. This thesis will provide an approach to fill the gap of *how to navigate huge amount of textual data*. This is done by answering the following research questions:

- How to organise textual data?
- How to make this data accessible ?
- How to present this data for navigation?
- How to evaluate the quality of such a system?

These questions are considered by presenting a concrete example that will serve as case study. This example is about a solution that is used to search for research data sets. Thus, the next section gives a brief introduction about the European data infrastructure project.

1.2 The European Data Infrastructure

The amount of data records generated by today's many research communities, as well as individuals, is increasing exponentially. This makes it increasingly complex and difficult to access, share or exploit this data for scientific research [61]. Due to this, many research institutes recommend their researchers to use data management solutions before starting new projects [14].

The European Data Infrastructure (EUDAT) is a project that comprises 25 European partners, including technology providers, research communities, and funding agencies from more than twelve countries. It works with multiple research communities and individuals to manage the rising tide of scientific data via advanced data management technologies [83]. The goal of this project is to provide European researchers who are producing or using very large data sets for research purposes with data services to create, maintain, archive, preserve, or find research data. At present, there are five data services that can be used by research communities.

- **B2DROP**: A Dropbox-like secure cloud storage dedicated to researchers to exchange, store, and keep their research data up-to-date. B2DROP allows them

to define the permissions, duration, and the method to exchange their research data with other researchers [30].

- **B2SHARE**: A reliable, user-friendly open solution for researchers and research communities to store and share small-scale research data in many extensions and formats. B2SHARE facilitates data storage, guarantees a long duration of persistence, and promises worldwide sharing. Researchers are able to define access policies for their data and allow it to be integrated using application programming interfaces (API) [33].
- **B2SAFE**: This service was developed to enable researchers implementing data management policy rules on their data across different regions. In addition, B2SAFE prevents this data from being lost in case of long-term archiving or preservation, and serves it for running on high-performance computers for intensive processing and analysis [32].
- **B2STAGE**: An extension of the B2SAFE and B2FIND services, it was developed to create a reliable and efficient way of transferring large-scale research data sets between high-performance computers and EUDAT storage resources [34].
- **B2FIND**: A data discovery service launched by EUDAT to find the research data stored by services such as B2SAFE and B2SHARE. This solution provides an interface to search and browse datasets via keyword searches and currently contains more than 420,000 indexed records based on metadata harvested from research data [31]. Furthermore, B2FIND provides a graphical user interface (GUI) to interact with its modules using search fields and facets.

Figure 1.1 shows the infrastructure provided by the EUDAT for researchers to maintain their research data.

As one of the most important platforms in research data management in Europe, EUDAT has been working with many research communities to determine their needs for handling research data, as it is trying to develop new relevant solutions and improve existing services [36]. Thus, this thesis uses B2FIND service as proof-of-concept to illustrate the various methods discussed for optimising the search and navigation of text data.

1.3 Research Scope

The contribution of this thesis is not intended to be the reimplementation or examination of B2FIND. The thesis will merely focuses on approaches to improve the user productivity when navigating big amount of textual data. This is done by:

1. studding the dominant approaches used in today's solutions
2. showing the drawbacks of these solutions
3. suggesting a model that uses graphical visualisations for text representations
4. suggesting a model that uses word clouds for text representations
5. integrating the suggested models with B2FIND as proof-of-concept
6. using classification and clustering approaches to categorise text data
7. evaluating the quality of the model using test users

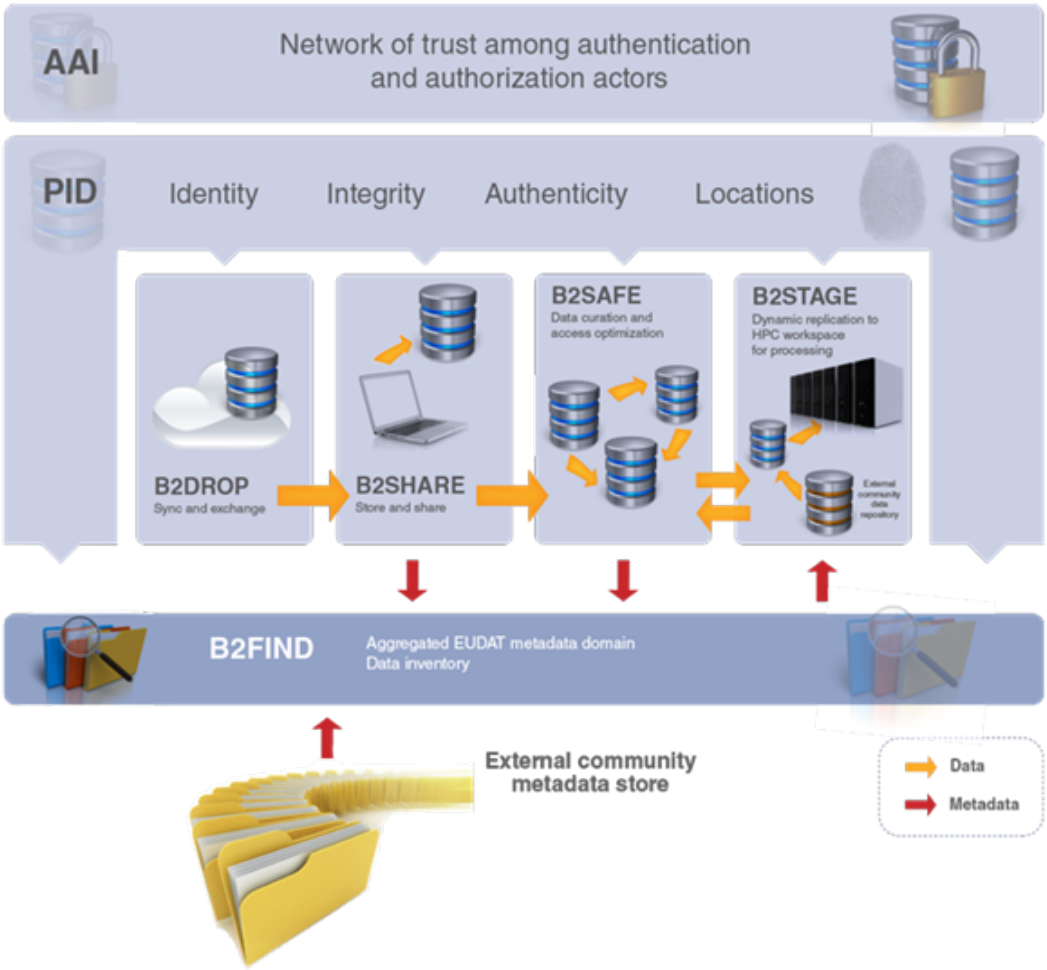


FIGURE 1.1: The European Data Infrastructure [31]

1.4 Thesis Structure

The thesis is organized as follows: The first chapter introduces the research, the research questions, the research context, EUDAT. Chapter 2 presents a literature review, shows the benefits of an optimised text navigation interface, proposes a new model to improve the issues of classical navigation models, and introduces B2FIND as case study. A theoretical discussion about text categorisation is also presented at the end of this chapter. Chapter 3 gives an overview of the experimental setup of this research. First, it introduces the architecture of the application. Then it shows analyses of the data from B2FIND and mentions the tools used for the implementation. In addition, it introduces an approach that we used to evaluate the methods proposed in Chapter 2. In Chapter 4, the results of the implementation of each interface and the results of the experiment are discussed in detail. Chapter 5 presents the final conclusion and suggestions for future research.

Chapter 2

Background

"*Many receive advice, only the wise profit from it*" - Harper Lee

2.1 Introduction

This chapter gives a theoretical explanation about all topics in this thesis, starting with explaining the importance of a good presentation model when navigating textual data. Then concepts of accessibility, usability, and user experience are described to understand their guidelines to build web interfaces. In addition, text navigation techniques that we used in our suggested solutions are introduced, as well as B2FIND as service proposed as proof-of-concept for the navigation strategies. We Specifics about the text categorization (TC) domain and a detailed explanation on how to evaluate the performance of the text classification systems and navigation strategies end this chapter.

2.2 Overview

Much research on search engines [55, 26, 51, 6] and database systems [8, 80, 72] focused on studying approaches which improve the quality of textual search results. It is assumed that only accurate search systems with high performance improve the search process. However, such systems are interfaced to a graphical representation that allows navigating the search results. Thus, their assumption is only valid if the adopted representation is optimal.

An interesting approach [2] divides the search and navigation process into three stages. In each stage, it shows the behavior and actions of the user (Figure 2.1 shows these stages and their relation [49, 3]). According to the authors, the last stage (*Information-retrieval*) is the most important because even if the tool used is powerful, the user could fail if he/she had no idea about the query he would start with; or when the results are not well presented.

- **Work task**: At this stage, the user recognises an information need and decides to satisfy it.
- **Information-seeking task**: At the second stage, the user selects a strategy to use in order to satisfy his/her need. In addition, the user chooses a tool for these tasks (for instance, using a search engine, navigating to a specific page etc).

- **Information-retrieval task**: At the last stage, the user formulates the needs in form of search queries, examines the result and selects a document. These three actions are repeated, until his objective is reached.

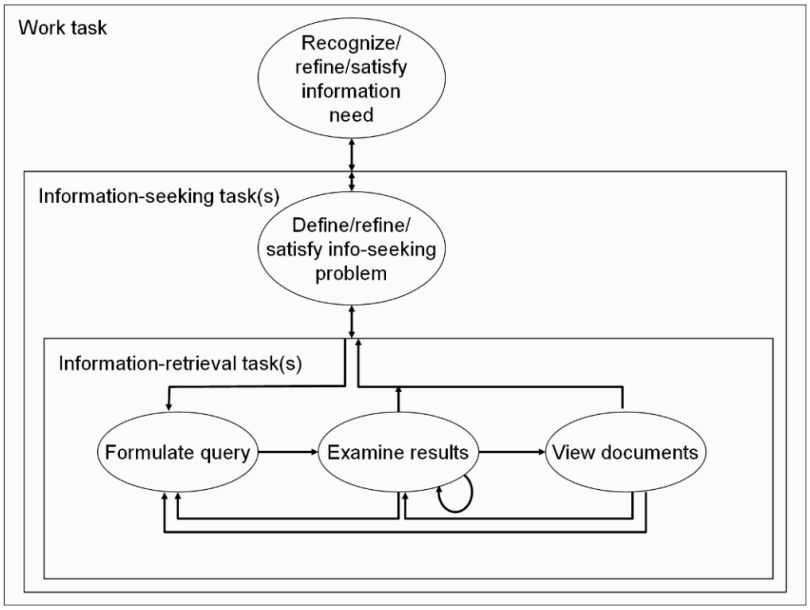


FIGURE 2.1: Search and navigation model [2]

Currently, the dominant approach for navigating textual data is a keyword text entry followed by the presentation of a textual list of results in relevance order [63]. This appraoch gained its popularity as s sufficient way to use the network bandwidth and screen space (a pagination is used to only retrieve a sub set of the data by each request). But in the present circumstances, the network bandwidth is not a real issue anymore. The real issue when navigating this data is the sheer amount: The search results become too long, thus a significant amount of screen space is required. In addition, it requires a lot of effort from the user to navigate that many pages (in most cases, the pagination contains many pages and is positioned at the bottom of the page).

Extensive research [3] has been done on optimising the search results, and dividing up the actions during the information retrieval into six actions. Each of these actions is termed according to a certain effort it requires. The first action is called *scan screen* and means that a user has to view the results on the screen and determine which of these results are most relevant. According to the author, the effort needed for this action is mostly medium, but could be minimised to a low effort if the result is categorised (for example, the result can be aggregated into categories of documents with similar content). The second action is *scrolling* (medium effort) and occurs when the screen is small, the design of the interface is bad, or too much information is presented. The author notes that scrolling is desired only in cases where the user is just reading (in case of weblogs), and not searching for an information. The third action is *pagination* (medium effort) and means that the result is divided into sub-sets of documents where the most relevant are placed on the first page. However, relevancy depends on the search query and algorithm used, and in some cases, the most relevant result could be on the second or third page of search

results. The fourth action is *item selection* (medium effort) and refers to the action of isolating the specific item from the result list. The fifth action is *reformulating the search query*. The author marked this action with a hight effort because the user has to think about strong keywords to find the most relevant documents that satisfy his information need. He also recommends to identify strong keywords in relation to the keyword given by the author and add them to the search query, in order to minimise the effort behind this action. The last action is *view web page* and has a variable effort as it depends on the design of the interface.

Table 2.1 summarises the actions that are usually done by a user when searching or navigating textual data. This table illustrates the effort of each action and mentions potential issues.

Action	**Effort**	**Issue**
scan screen	medium	mixture of documents
scroll screen	medium	pages too long
pagination	medium	too many sub-pages
select result	medium	relevant documents, duplicate pages, mixture of documents
reformulate query	high	search keyword to start with
view page	variable	depends on the design

TABLE 2.1: Issues and effort of actions in classical interfaces [2, 3]

2.3 Accessibility, Usability, and User Experience

After having discussed the importance of the interface when searching for textual data, seeing the dominant approach used in today's solutions and their issues, we suggest a new navigation model that provides improved components, and especially offers new ways to search for documents and navigate text data. But before we do that, we would like to introduce some concepts in accessibility, usability, and user experience (UI).

2.3.1 Accessibility

Accessibility refers, in our context, to a measure of how obvious and easy it is to access, navigate, and understand the content of a web page. We have to note that some users may be operating or processing some types of information differently than others. They could even have old versions of web browsers. Tim Berners-Lee[1] defined accessibility as the possibility of accessing a page from everyone, including people with disabilities: "The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect". Thus, it is required to follow some simple guidelines [12] that explain how to make web content accessible. In general, these 14 guidelines address two aspects: making content easy navigable and making it understandable.

1. Provide equivalent alternatives to auditory and visual content.
2. Don't rely on colour alone.

[1] **W3 Director and inventor of the World Wide Web https://www.w3.org/standards/webdesign/accessibility**.

3. Use markup and style sheets and do so properly.
4. Clarify natural language usage.
5. Create tables that transform gracefully.
6. Ensure that pages featuring new technologies transform gracefully.
7. Ensure user control of time-sensitive content changes.
8. Ensure direct accessibility of embedded user interfaces.
9. Design for device-independence.
10. Use interim solutions.
11. Use W3C technologies and guidelines.
12. Provide context and orientation information.
13. Provide clear navigation mechanisms.
14. Ensure that documents are clear and simple.

2.3.2 Usability

Usability refers to the extent to which users can accomplish a specific task [7], it measures the ease with which a user can operate, interact with input and output of a system or component [39]. In the context of this thesis, we use the definition provided by [75], that refers to the usability as: "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". In so doing, each time we were working on the prototypes, we asked the three questions:

- **Effectiveness**: How accurate could a user achieve certain task?
- **Efficiency**: How many steps did the user take to achieve the goal?
- **Satisfaction**: How comfortable is the work system to its user?

2.3.3 User Experience

User experience was defined by [76] as the perceptions, emotions, beliefs, and preferences, that result from the use of a system. Based on this definition, we can consider the usability as part of user experience. This judgment was supported by [24], who considered user experience as a consequence of three factors:

1. **Characteristics of system**: comfortability, functionality, usability.
2. **User' s internal state**: motivations, expectations, tasks.
3. **Context**: content, organization.

Figure 2.2 shows the differences and relationships between usability and user experience.

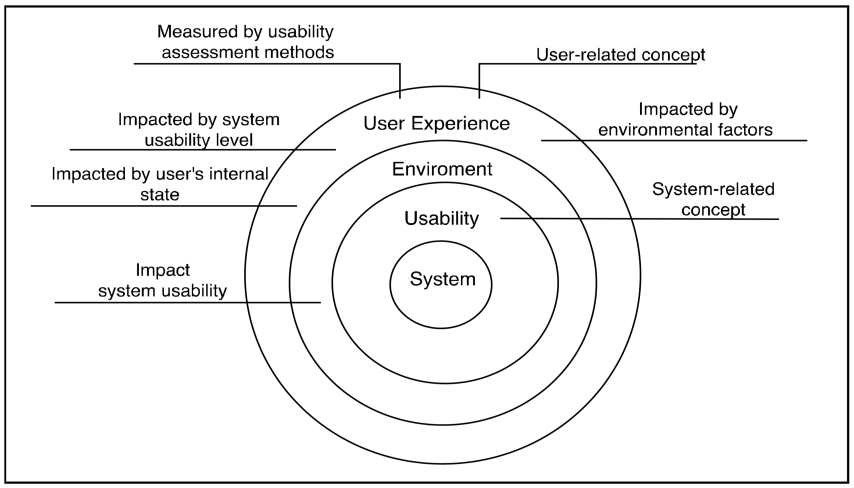


FIGURE 2.2: Differences and relationships between usability and user experience [22]

2.4 Navigation Model

As we said in the previous section, the aim of this thesis is to propose various ways to navigate textual data. Each of these methods has to be according to the guidelines, and minimises the search effort of the actions discussed in the previous sections. As an evaluation method, an approach is used that is discussed in the next chapter.

2.4.1 Information Architecture

According to [57], information architecture is the science of structuring information to support its usability and maximise its accessibility. In our context, we refer to it as the way that the data has to be transformed in order to be understandable. Thus, the first step is to break up the page into four sections, and then discuss the type and kind of information that should be presented in order to increase the usability and user experience.
Figure 2.3 shows an example of the proposed structure. The areas marked in colors indicate the four components we mention before.

Blue Area

This section is called header and is usually the first section seen by a user. Therefore, it contains representative and control components, and has to be static and informative. Four our design, we suggest to provide content that helps the user answer the following questions:

- Where am I?
- What should I do to find what I'm looking for?

The answer to the first question can be delivered using textual or visual content (e.g. using a description, image, or logo etc.). Here, the user wants to ensure that he/she is using the right service. In order to answer the second question, the user expects some component that assists to find what he/she is looking for. Thus, we

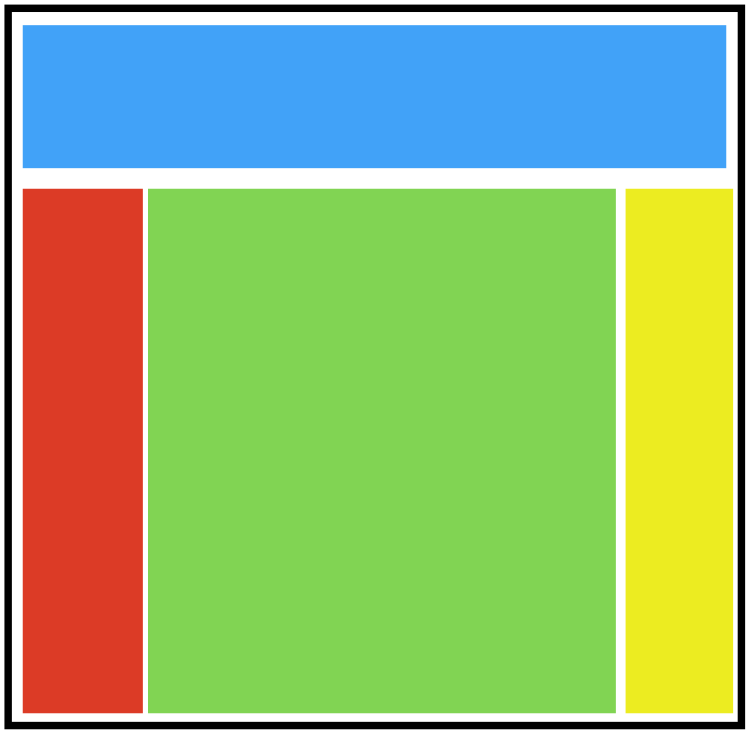


FIGURE 2.3: An example of system layout proposed to structure the data. Each color represents a section

use a search field that serves as a controller to search for data, and trigger the content displayed in the other sections (see Example 2.4).

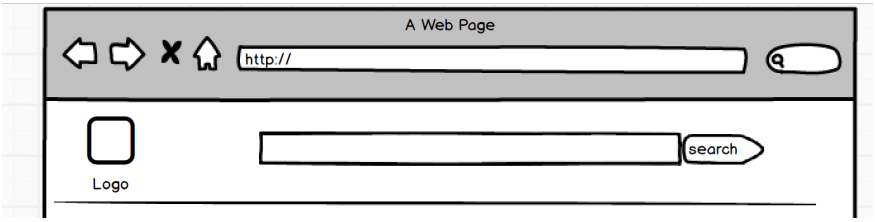


FIGURE 2.4: An example of header that contains a logo in the left side and a search field in the right side

Green Area

The green section is the main and most important. It contains the search results after using the search field discussed previously. If it is not user friendly and understandable, the user cannot succeed.
At the initial state, the search field is mostly empty; in this case, the content of this section depends on the strategy followed:

- **Strategy 1**: We just leave the section empty and inform the user to start navigating by using the search field presented in the previous area. This strategy is simple and easy to implement, but has a drawback when the user doesn't know the search term to start with.

- **Strategy 2**: We can display a list of documents that the user may use to start navigating. These documents could be selected randomly or based on a specific criterion. The problem here is to define criteria that suit every user. Another drawback is that in some cases, the user has to paginate many pages until finding what he/she is looking for.

As we mentioned earlier, when a user searches for a document using a search term, in many cases the result found is very long. Thus, a pagination is often used to break up a set of documents into subsets in order to put them in pages. But some paginations can have too many pages, which leads the users to not browsing them all [54]. Another issue with the results is the possible presence of duplicate content. This could decrease the diversity of results shown on each page, and also force the user to either visit each document to explore its content or, in the worst case, to quit to search.

We want to avoid reimplementing similar solutions to those we discussed in section 2.2, but at the same time we want to deliver a better way to navigate documents. If we use a list to display the results, we face the same issues (for instance, many duplicate documents or long result lists). Another issue that we want to solve is the mixture of results, such as mixing documents from various research areas. Thus, we suggest to cluster the result in groups of documents that share the same research area, discipline, and content. Figure 2.5 shows an example of clustered documents implemented by the **SCImago Journal**[2] community. The advantage of this representation is that the user would stay in the scope of his interest by searching for documents inside a cluster only. However, it has the drawback that the user would have to hover over each document to see its name. To solve this drawback, we suggest to use of a word cloud representation instead. The workflow is as follows:

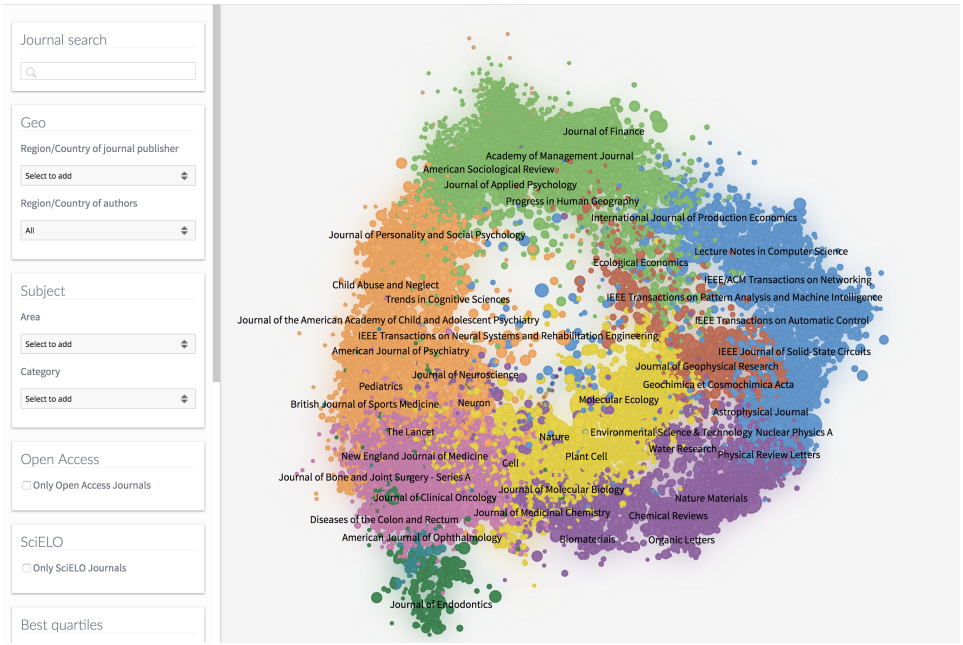


FIGURE 2.5: An example of header documents representation by Scimago

[2] **https://www.scimagojr.com/shapeofscience/**.

1. When the search field is empty, we show the top keywords of each cluster. The user is able to select the desired number of clusters he/she wants.
2. When the user inserts a search term, we look for this term in each cluster and display a result in the form of a word cloud.
3. For each given term, we look for documents in all clusters that contain these terms and display only clusters with hits.
4. If the user selects a word from a cluster by clicking on it, we only continue looking for documents inside that cluster.

Figure 2.6 shows an example of a word cloud. The size of each word symbolises its importance (number of occurrence). The advantage of this method is that we can summarize words of duplicate documents by increasing their size and emphasis strong keywords (recall the high effort behind finding or reformulating the search query), although this solution unfortunately has a negative effect when there are a large number of words inside each cluster. The representation would be unclear and the user can be confused.
In order to avoid this negative effect, we defined some rules:

- We only show the top N keywords based on their importance.
- We remove prepositions, articles, pronouns, adjectives from the result because they do not relate to the content (see Appendix A Table A.1).
- We know that the result found is based on the given search terms, therefore we hide them from the result list.
- If a keyword appears with a specific importance in many clusters, we declare it as insignificant and remove it from the results. The reason is that in such a case, the keyword is meaningless and uninformative concerning the content of the clusters. In addition, the user would have to explore each cluster having that keyword.

We mentioned before that when a word is selected from a cluster (see workflow discussed above), we only look for documents that match the given search terms, including this word inside its cluster. In this case, the list of results will also be grouped in clusters. There are various ways to tackle this navigation problem:

- **Multiple Clustering Navigation Approach**: This approach is illustrated in Figure 2.8. In this example, a user would see the collection of documents clustered into two groups; if he selects a word from cluster one (blue), he would only get documents that match his selection, and the result would be grouped in three sub-clusters. By choosing a new keyword (for instance, from the sub-cluster on the right side), the user would again get only matching documents, this time clustered into two groups, etc.
  The benefit of using this approach is that it gives the user a general idea about the scope of documents in each cluster. However, multiple clustering navigation needs a high calculation performance and long processing time when building the clusters. Moreover, its quality depends on the clustered data.
- **Hierarchical Navigation Approach**: After clustering the collection of documents into groups, the result of each sub-cluster is then grouped by its meta-data. An example of such a navigation technique is illustrated in Figure 2.9. In

FIGURE 2.6: Example of word cloud [85]

this example, if a user selects a word from a cluster, the result is grouped into disciplines (for instance programming languages, network, security...). By selecting a new word from a discipline, the result will match documents grouped by its authors, etc.
The first advantage of this approach is that we avoid building clusters each time. The second advantage is that the data is grouped based on metadata given manually, and does not relying on calculations based on data that could be of bad quality. However, it is important to choose a grouping variable without many different unique values, otherwise the group size would be too long. This approach could also be confusing when the user has no idea about the context and values of the grouping variable (for instance, the user has no idea about an author or his research area).

Red Area

This area is designed to contain elements needed to filter the result, and to control the size of clusters and their content. In this area, we put elements that give the user the ability to:

- select the number of clusters to start with.
- only select words that only appear in one cluster.
- filter the result based on the document 's metadata (for instance, return documents from a discipline, of an author or published within a certain date range). However, this is optional as the metadata are not always available.

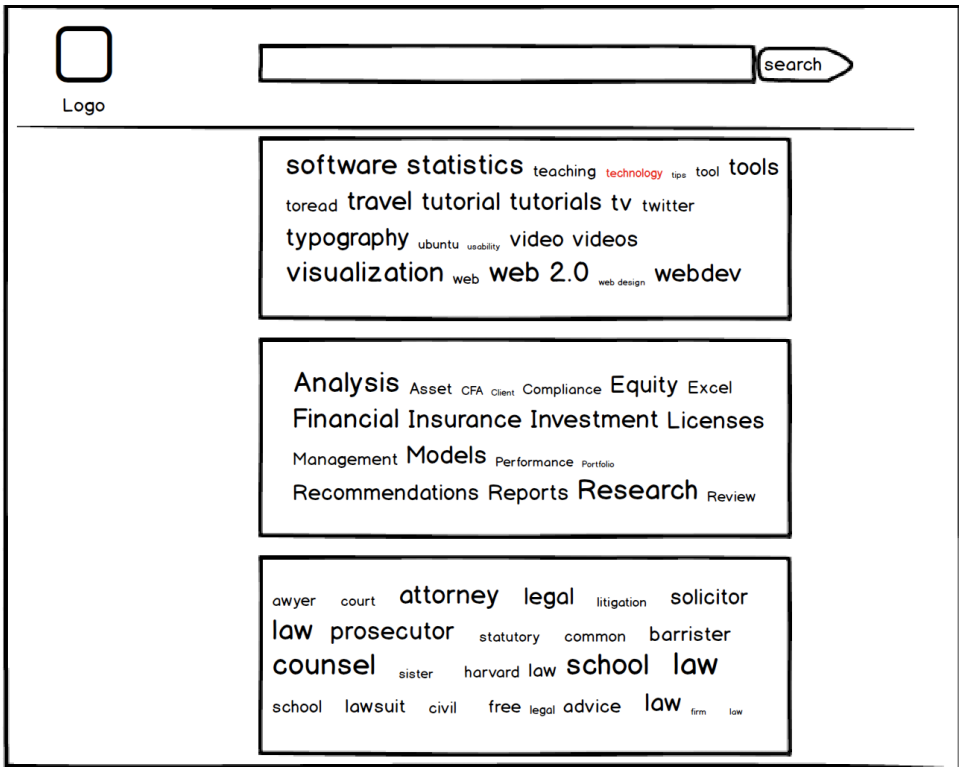


FIGURE 2.7: Example of a representation based on word cloud

Yellow Area

In this section we would like to display the top N documents of the result set. There are two approaches that can be used in this case:

- Retrieve the top N documents from the collection set based on the search query. This approach is easy to implement but suffers the drawback that in some scenarios it only includes documents from one cluster in the top N list.
- Get the top N documents from each cluster. An advantage of this appraoch is that we assure displaying top documents from each cluster based on the search query.

Figure 2.10 illustrates an example that summarises the approaches discussed in this section.

2.5 Case Study: B2FIND

B2FIND is an EUDAT discovery service designed, developed, and maintained by the Deutsches Klimarechenzentrum (DKRZ). It is a central catalogue that uses metadata gathered from research data collections and external community centre archives, such as B2SHARE, to allow users to easily find scientific research data from different communities [35].

The objective of B2FIND 's developers was to design a solution that is modular, in order to avoid technology lock-in and to be flexible when adding or changing new components. This is also the reason why they decided, after many evaluations, to

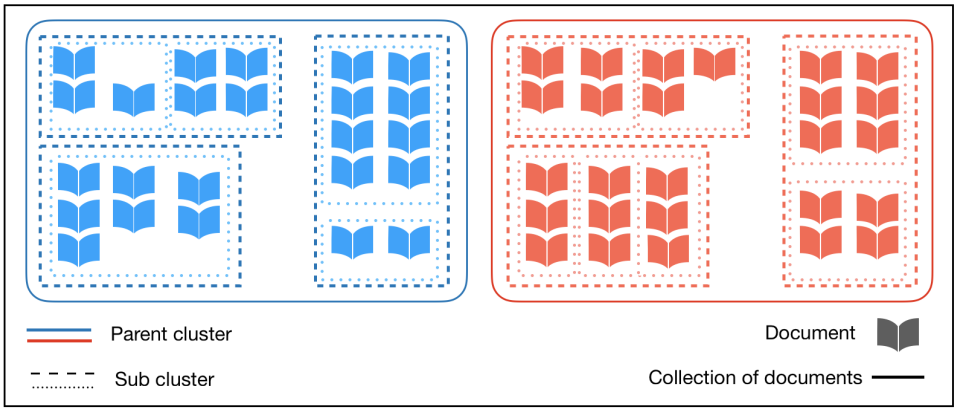


FIGURE 2.8: An example of multiple clustering navigation approach. The documents are grouped into two clusters (blue and red), each cluster is clustered again into sub clusters

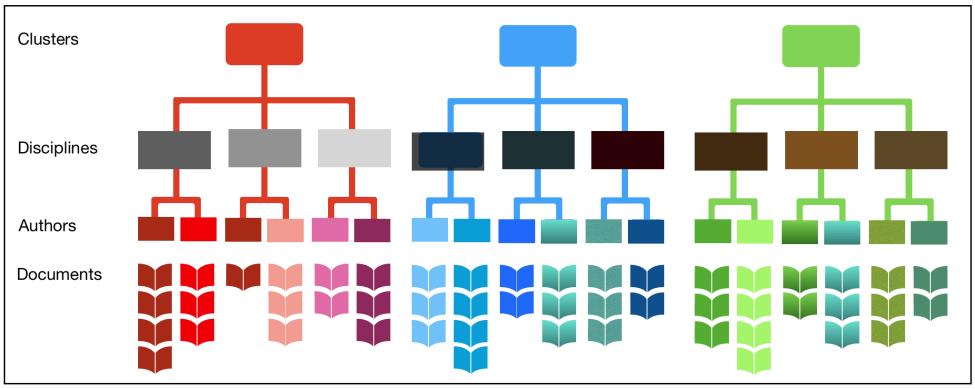


FIGURE 2.9: An example of hierarchical navigation approach. The documents are grouped into clusters (red, blue and green), and then inside the clusters into disciplines, and finally by authors

use CKAN[3] as the underlying technology [69]. CKAN is an open-source data portal platform that is widely used for publishing, sharing, and finding data.

Figure 2.11 shows the architecture of B2FIND, which consists of **harvesters** that collect metadata records (Table 2.2) from data collections and external archives, such us OIAI-PMH[4], **mappers** to transform this metadata to the JSON[5] format, and an **uploader** to import the transformed data to B2FIND repositories. CKAN receives these data over APIs and index it to **SOLR**[6].

Currently, the only way to access B2FIND is to use its graphical user interface with a standard web browser (Figure 2.12). This interface offers a search field for keyword search and a facet of filters. Table 2.3 illustrates a list of variables used to filter the search result. [74] conducted several tests to evaluate the usability and user experience of EUDAT, based on questionnaires and scenarios where test users were given tasks to complete. For B2FIND, the users had to complete twelve tasks, such

[3] **https://ckan.org/.**
[4] **OAI-PMH stands for Open Archives Initiative Protocol for Metadata Harvesting.**
[5] **JSON stands for JavaScript Object Notation and is a lightweight data-interchange format. See https://json.org.**
[6] **SOLR is highly reliable, scalable and fault tolerant search machine.**

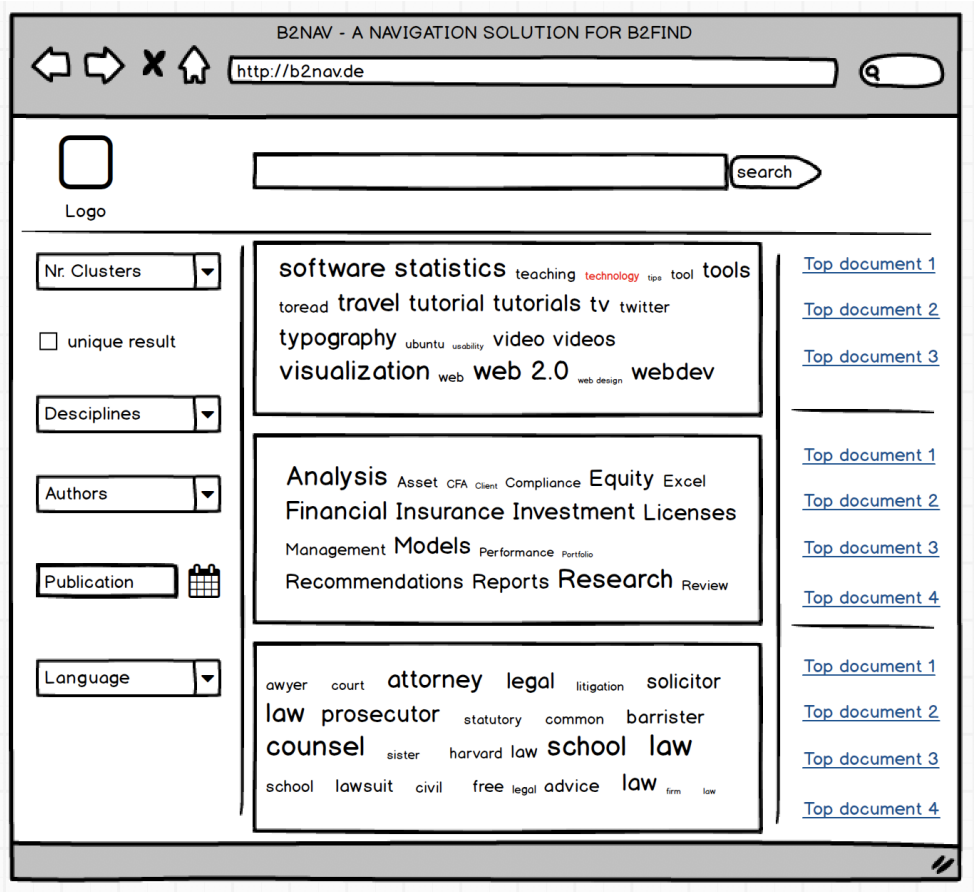


FIGURE 2.10: A final example of suggested system

as doing some searchers using special keywords, filtering the results, checking the metadata, and loading the data sets. The result of his evaluation showed that many users were unable to complete their tasks without help, or the task took more than the average time due to the difficulty of some components.

- **Time**: The time filter is confusing, the date formats suggested to assist the user by typing a start and end period are different; the end field, for instance, shows a date with time, whereas the start field shows a negative random value. In addition, a system window appears when the user clicks on filter by time.
- **Creator**: This filter allows a user to select multiple available options and it has a search field to look for creators. In some cases, the returned name is too long to be displayed entirely; thus, B2FIND truncate the creator name to a specific length that makes the result ambiguous (Figure 2.13).
- **Resource Type**: This search filter has two issues; the first is about the suggestions displayed to the user, that are in some scenarios unclear (Figure 2.14 left capture); and the second issue appears when clicking on one of these suggestions (Figure 2.14 right capture).

Generally, the main issue with B2FIND 's filters is the inconsistency of all components. Some fields are applied directly after selection, and others need a click of the apply button. In addition to that, the result listed in the middle of the page is in

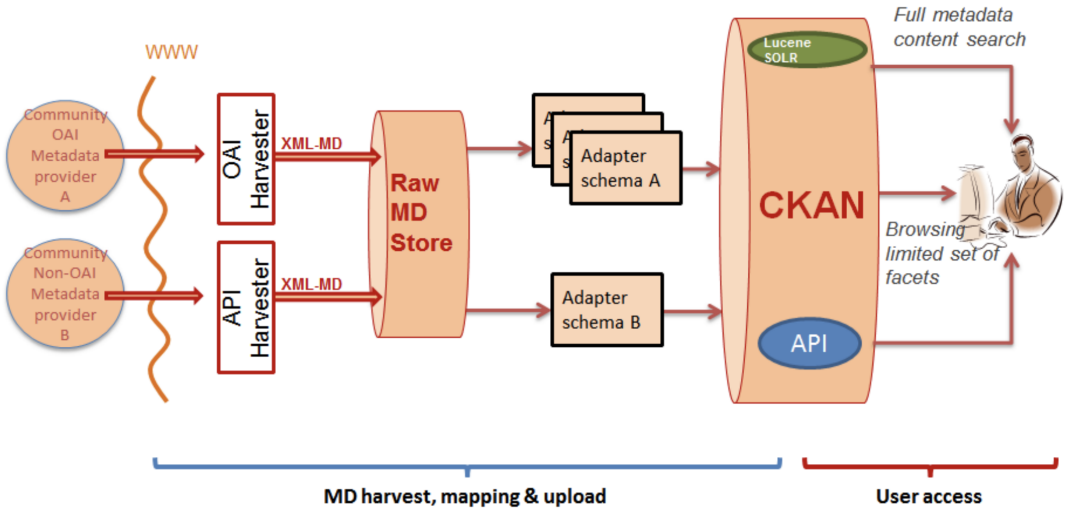


FIGURE 2.11: Overview of B2FIND's architecture [69]

Variable	Description
Title	The title of the uploaded document that indicates the content to be expected.
Author	The name of the author(s).
Url	An external URL of the resource explained in the document.
Notes	A note about the uploaded resource.
Tags	A list of tags about the resource.
Group name	The name of the group that published the resource.
Group title	The title of the group that published the resource.
Group description	A description about the group that published the resource.
Group image	A logo of the group that published the resource.
Extras	This fields contain many values such as the language, the publication date, and the rights of the resource.

TABLE 2.2: A list of metadata stored in B2FIND

some scenarios useless and uninformative. For instance, the search result of **physical oceanography** contains many duplicate and redundant items that negatively affect the productivity of users, and even could drive them away from using the service (see Figure 2.15).

2.6 Text Categorisation

As established in the last section, the solution suggested in this thesis is based on grouping documents that contain textual data into clusters. In order to do that, we started using classification approaches to generate these groups. However, the result was not satisfactory, therefore we redesigned our solution to use clustering methods. In this section, we discuss various approaches.

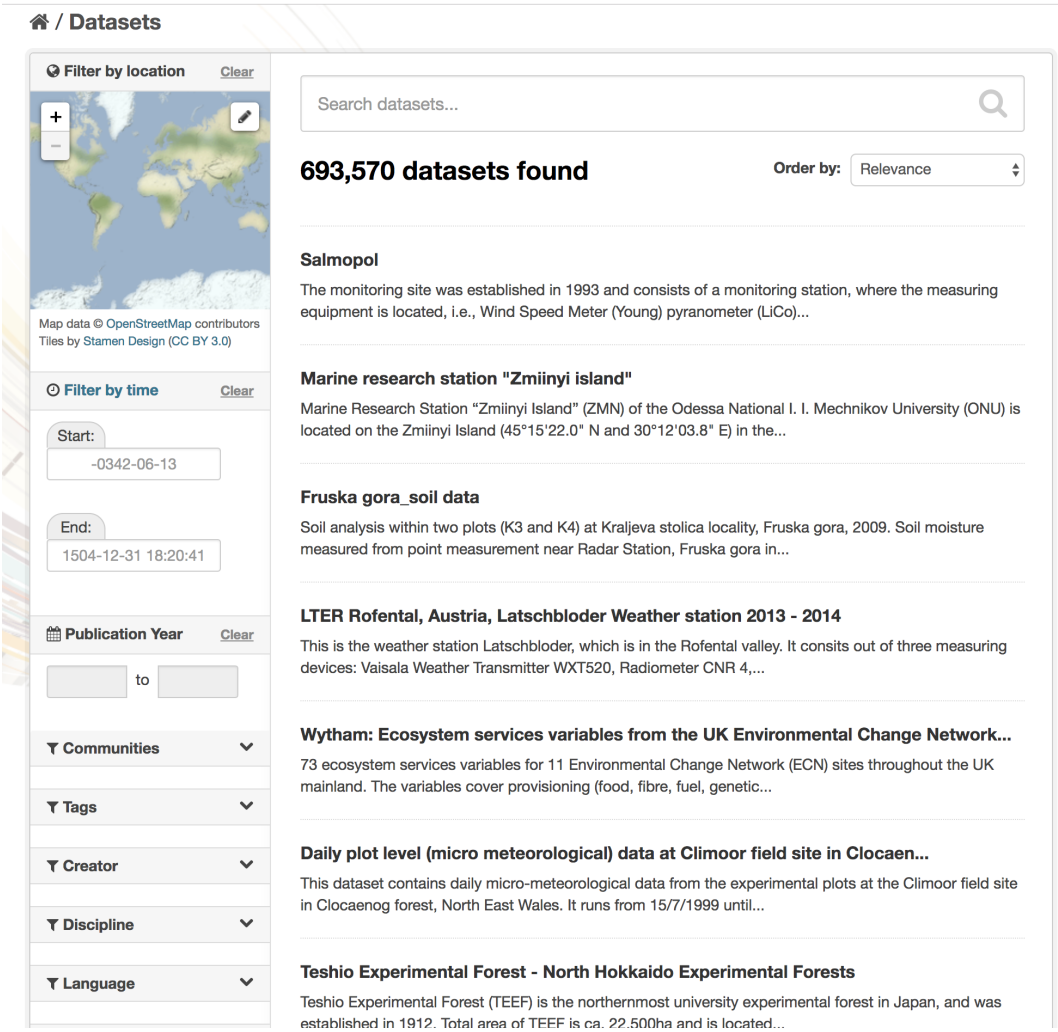


FIGURE 2.12: A screen capture of B2FIND 's graphical user interface [31]

2.6.1 Text Classification

Generally, classification is the task of assigning objects into one or more predefined categories called classes or labels [86]. Classification problems belong to supervised learning problems (definition 2.6.1 [53]) where the learner tries to output a value that match an object to a class.

Definition 2.6.1 ***(Supervised Learning)***

The goal of supervised learning is to seek a function $g : X \longrightarrow Y$ *which maps an example* $x \in X$ *to its output value* $y \in Y$.
$X = \{x_1, x_2, ..., x_N\}$ *is the input space, where* $x \in X$ *represents the input object in the data.* Y *is the output space, where* $y \in Y$ *represents an output value from the possible values in* Y. *The learning problem is called regression learning if* $Y = \mathbb{R}$, *and classification learning if* $Y = C$, *where* $C = \{c_1, c_2, ..., c_M\}$ *represents the set of possible classes.*

An example of a problem from definition 2.6.1 applied to a text classification task can be a set of products that have to be classified to predefined categories based on their description. A classifier that assigns a value from the set $C = \{laptops, tablets\}$

Filter	Description
Time	A time range field to set the start and end date of the data set.
Publication Year	The year when the research was published.
Communities	A list of names of existing communities.
Tags	A note about the uploaded resource.
Creator	The institute that created the resource.
Discipline	The discipline of the resource (extracted from the field *Extras* - see Table 2.2).
Language	The language of the resource (extracted from the field *Extras*).
Publisher	The publisher of the resource (extracted from the field *Extras*).
Resource Type	This field contains many values such as dataset, series, survey data (extracted from the field *Extras*).

TABLE 2.3: A list of options used by B2FIND to filter the search result

is called *binary classifier*. If the classifier assigns a value from a set C that contains more than 2 values (for instance $C = \{laptops, tablets, printers\}$), then the classifier is called a *multi-class classifier*. When the classifier assigns a set of target labels to a single product, the classifier is called *multi-label classifier*.

Two components are important to be designed carefully in order to build up a good classification system. The first component is the input that has to be transformed and preprocessed before it is provided to the classifier. The second component is the function g (the classifier itself) that has to be appropriately selected depending on the input data [20]. In the next chapter, we will discuss various methods to convert the data to a set of features that can be used by the classifiers for text classification tasks.

2.6.1.1 Pre-Processing

Before textual documents are used for classification problems, some pre-processing tasks with significant impact on the classification process are usually performed. One of these important tasks that need to be accomplished is that of document preprocessing, document representation, and feature selection.
The document pre-processing stage can consist of many stages such as dividing the document up into pieces of words or phrases called tokens [82], removing insignificant words which frequently appear in the text without having much content information [68] (e.g. prepositions, conjunctions, etc, see the stop words list in Appendix A), and stemming words in order to convert them into their similar canonical form [29] (for instance, the word computing will be converted to compute, lovely to love etc.).
Table 2.4 shows an example of a document (original text) and its output after passing each stage. Note that it is not required to use all these stages in the pre-processing phase. The designer of the classification system has to choose the appropriate transformation based on the kind of data and quality of the result (for instance, the phrase "runs yesterday" may refer to a runner, while "long running" may be about a computer battery lifetime. In this case the stemmer makes the classification quality worse, not better).

FIGURE 2.13: A screen capture of B2FIND 's *Creator* filter. The 5-th row of the result is truncated after 16 characters which led to ambiguity problems

2.6.1.2 Feature Generation

The next step after pre-processing the documents, is to convert their data in a representation with which a classifier can work (this representation is called **Vector Space Model** (VSM), see Definition 2.6.2 [20]). In general, textual data can be represented in two methods: The first method represents the text as **strings**, in which a document is a set of words [9]. The second method is called **bag-of-words** and is the most common way to represent textual data, because of its simplicity for classification purposes [43, 50, 20].

Bag-of-Words Model

The bag-of-words (BOW) model considers each document as a set of words (called *features* or *terms*) that occur with a certain frequency. This representation of the document is entirely independent of the sequence of words in the collection. As an example, we have a data set (Table 2.5) consisting of only two documents:

Table 2.6 shows the BOW representation of the documents of Example 2.5. Each document d_j is represented as a vector $\vec{d_j} =< f_{1j}, f_{2j}, ..., f_{tj} >$ where f_{ij} is the number of occurrence of a feature in the document d_j.

Definition 2.6.2 ***(Vector Space Model)***

Given a collection of documents $\mathcal{D} = \{d_1, d_2, ..., d_{\mathcal{D}}\}$, *let* $\mathcal{V} = \{v_1, v_2, ..., v_{\mathcal{V}}\}$ *be the set of distinct words in the collection. Then* $\mathcal{V}$ *is called the vocabulary. The frequency of the word* $w \in \mathcal{V}$ *in document* $d \in \mathcal{D}$ *is shown by* $f_{w,d}$. *The term vector for document* d *is denoted by* $\vec{d_j} = (f_{w_1,j}, f_{w_2,j}, ..., f_{w_{\mathcal{V}},j})$.

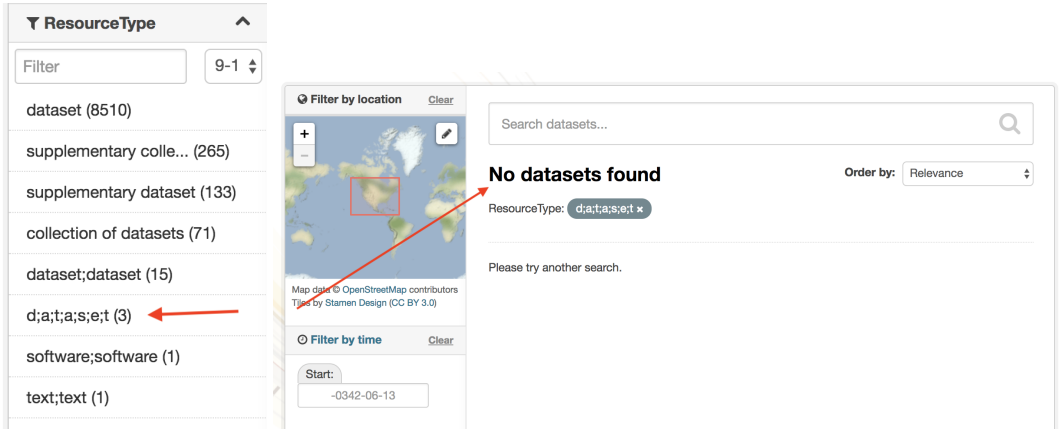


FIGURE 2.14: A screen capture of B2FIND 's *resource type* filter. The left figure shows some invalid suggestions, such as in the 6th row. The result list is empty, although the 6th row shows 3 existing data sets

Stage	Function	Output
1	Original text	A document is a written, drawn, presented, or memorialised representation of thought.
2	Lowercase	a document is a written, drawn, presented, or memorialised representation of thought.
3	Remove stop words	document written drawn presented memorialised representation thought
4	Stemming	document written drawn present memori represent thought
5	Tokenizing	*list*('document', 'written', 'drawn', 'present', 'memori', 'represent', 'thought')

TABLE 2.4: An example of a text pre-processing workflow

A problem of the BOW representation is that much information from the original document is discarded. The sentence and word order is disrupted, and syntactic structures are broken [50]. For instance, in a sentiment analysis classifier, negations of words needs to be taken into consideration because of their appearances that often change the opinion orientation (e.g. the sentences "I like to watch movies" and "I don't like to watch movies" have different meanings. If we remove the term *don't* in the preprocessing stage, or do not combine the terms *don't* and *watch* together as feature, we would classify the sentences as positive).

N-gram Model

Many researches [71, 1, 17, 50] have been done on using phrases as terms in the BOW approach. These studies analysed the usage of a sequence of words instead of a single word when generating the features. This model is called n-gram, where n denotes the number of words used. Table 2.7 shows the 2-grams of the documents in Example 2.5.

Generally, the usage of the n-gram model is not clear and can negatively affect the accuracy of the classifier. If we choose a big number n, the number of features will increase very fast and leads to sparse vectors d_j.

Document one	The cat is faster than the dog
Document two	The dog is in the house

TABLE 2.5: Example of a data set

	the	cat	is	faster	than	dog	in	house
Vector one	2	1	1	1	1	1	0	0
Vector two	2	0	1	0	0	1	1	1

TABLE 2.6: Bag-of-words feature example

TF.IDF Weighting

Another problem of the BOW representation is that the usage of the frequency of a word does not give its importance for a document. For instance, the term *the* of Example 2.6 can appear in every document but does not give enough insight as the term *cat*. TF.IDF is the most popular term weighting scheme [64] that provides a relevance (weight) of a term for a particular document. It combines the term frequency (TF) with the number of documents containing this term (document frequency (DF)). The document frequency is proportionally inversed (IDF), due to the reason that terms which rarely occur over collections of documents are valuable [4].

Definition 2.6.3 $(TF.IDF(f_k, d_j, \mathcal{D}) = TF(f_k, d_j) \times \log \frac{|\mathcal{D}|}{DF(f_k)})$

where $TF(f_k, d_j)$ *denotes the TF of term* f_k *in document* d_j *and* $DF(f_k)$ *denotes the DF of term* f_k.
$|\mathcal{D}|$ *is the number of documents in the collection* $\mathcal{D}$.

Table 2.8 shows a calculation example of TF.IDF based on Example 2.5. We have two documents where the words *this* and *dog* appear. Thus, $IDF(this, \mathcal{D}) = \log\left(\frac{2}{2}\right) = 0$. The TF.IDF then is zero, which means that in these documents, these words are not very informative.

2.6.1.3 Classification Techniques

Now, the documents are pre-processed and the matrix containing the vocabulary and their weights for each document is calculated for each document (see Table 2.9). The next step would be to use a classifier to categorise these documents. We mentioned before that we will explain algorithms from the supervised learning approach first. Such algorithms assume that the category structure of the document is known. They rely on training data to learn from in order define a function that maps documents to the pre-defined class labels. For example, the matrix in Table 2.9 would have a sub-set of documents with their classes (typically a sub-set with 75% of this data for training, and 25% for tests), the algorithm would then learn from this sub-set and be applied to classify new documents.

Various supervised algorithms have been described in the literature on machine learning. Some of them are margin classifiers such as *Support Vector Machines* (SVM) [38], others are based on probabilistic approaches [66], such as *Naive Bayes* (NB) [52]. In this section we discuss the most popular supervised algorithms that we used in our practical part of this thesis.

the_cat	cat_is	is_faster	faster_than	than_dog	dog_in	in_house
the_dog	dog_is	is_in	in_the	the_house		

TABLE 2.7: An example of 2-grams generated from Example 2.5

	the	cat	is	faster	than	dog	in	house
TF_{d_1}	2/7	1/7	1/7	1/7	1/7	1/7		
TF_{d_2}	2/6		1/6			1/6	1/6	1/6
IDF	log(2/2)	log(2/1)	log(2/2)	log(2/1)	log(2/1)	log(2/2)	log(2/1)	log(2/1)
$TF.IDF_{d_1}$	0	0.043	0	0.043	0.043	0	0	0
$TF.IDF_{d_2}$	0	0	0	0	0	0	0.050	0.050

TABLE 2.8: A calculation example of TF.IDF weighting

Naive Bayes

Naive Bayes is a simple probabilistic classifier that is highly popular as supervised learning technique to classify textual data. It uses the joint probabilities of features and classes to estimate the probabilities of these classes in a test document [62]. This approach is called "naive" because of the assumption of conditional independents between features given class. Due to this assumption, the order of the features become irrelevant, which means that the present of one feature does not affect other features in classification tasks. Another benefit is that the parameters for each feature can be learned separately, and this positively influences the speed of computation as compared to other classifiers [40].

Classification tasks using the NB approach is applied based on the Bayes' rule

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)} \tag{2.1}$$

where d_j is a document and c_i is a class. In the classification phase, the posterior probability $P(c_i|d_j)$ for each class is calculated, and the highest probability is assigned to the document d_j. Note that the prior probability $P(c_i)$ can be estimated (we refer to it as $\hat{P}(c_j)$) from the training set as follows:

$$\hat{P}(c_i) = \frac{n_i}{N} \tag{2.2}$$

where N the is number of training documents and n_j is the number of documents assigned to the class c_i. The priori probability $P(d_j)$ is the same for each class therefore we eliminate it from Equation 2.1.
Because of the assumption we mentioned earlier, the documents will be drawn from a multinomial distribution of features with number of independent trials equal to the length of the document d_j. Hence, we can replace the class prior probability $P(d_j|c_i)$ by:

	f_1	f_2	f_3	f_4	f_5	f_6	...	f_v	class
d_1	0	0.337	0.271	0.512	0.103	0		0.015	c_1
d_2	0.095	0.125	0.703	0	0	0.373		0.023	c_3
d_3	0.3123	0.445	0.503	0.073	0.211	0		0	c_1
d_4	0	0	0	0.343	0	0.203		0.103	c_2
...									
$d_\mathcal{D}$	0.413	0	0.115	0	0.221	0.102		0	?

TABLE 2.9: An example of a TF.IDF matrix. The class of documents d_1, d_2, d_3 and d_4 is known. But the class of document $d_\mathcal{D}$ has to be predicted

$$P(d_j|c_i) \approx \prod_{k=1}^{|d_j|} P(f_k|c_i) \tag{2.3}$$

where $|d_j|$ denotes the number of features in document d_j, and f_k is the k^{th} feature from document d_j.
The class prior probability of the features $P(f_k|c_i)$ can be approximated from the training data, as we did before with the prior probability:

$$\hat{P}(f_k|c_i) = \frac{\alpha + TF(f_k, c_i)}{\alpha|\mathcal{V}| + \sum_{h=1}^{|\mathcal{V}|} TF(f_h, c_i)} \tag{2.4}$$

where $TF(f_k, c_i)$ is the total number of times the feature f_k occurs in documents of class c_i in the training set and $|\mathcal{V}|$ represents the size of the vocabulary as discussed in Definition 2.6.2. The constant α is a smoothing constant used to handle the problems of overfitting and the case where $TF(f_k, c_i) = 0$. Note that Equation 2.4 is called *Laplace estimator* if the constant α is equal to 1.

During the decision phase, the algorithm calculates the posterior probabilities of the document d and the set of classes c, based on Equation 2.5. In case of a binary classifier, the algorithm would decide for one of the both classes, based on their probability. In a multi-class classifier, the algorithm would take the class with the highest posterior probability $arg\ max\ P(c|d)$. A multi-label classifier would return the top-N classes ordered by the highest posterior probability.

$$P(c_i|d_j) \approx \frac{n_i}{N} \prod_{k=1}^{|d_j|} \frac{\alpha + TF(f_k, c_i)}{\alpha|\mathcal{V}| + \sum_{h=1}^{|\mathcal{V}|} TF(f_h, c_i)} \tag{2.5}$$

Support Vector Machines

Support Vector Machines (SVM) is a classification technique introduced by Vapnik in [81] and extended for performing text classification tasks by Joachims in [37]. It is based on Structural Risk Minimization principles and designed to solve two class pattern recognition problems using quadratic programming techniques [81,

48].
The conceptual structure of SVM is shown in Figure 2.16, based on an example of points in a two-dimensional feature space. The data points in this figure represent term weights from documents of two classes (negative samples are documents from the first class, and positive samples from the second class). The objective of SVM is to find the optimal hyperplane which separates the positive examples from the negative examples in the hyperspace. For instance, the solid line colored in red is a decision surface that separates the two classes, and the dashed lines show how much the decision surface can be moved without miss-classifying the documents. The problem of finding the optimal hyperplane is solved by calculating the decision surface that maximizes the *margin* [37, 42]. In our example, the decision surface is defined as:

$$\mathbf{w}\ \Phi(x_i) + b = 0 \tag{2.6}$$

where x_i is the input to be classified, and Φ is a mapping function on the data point (a function that maps the input space $\mathbb{R}^m$ to a feature space $\mathbb{R}^n$). The vector $\mathbf{w}$ that defines the orientation of the hyperplane, and the offset b ($b \in \mathbb{R}$) are learned from the training data. For example, if $D = \{(x_i, y_i)\}$ denotes the training data set that has N samples, whereby $y_i \in \{+1, -1\}$ represents the negative and positive examples, the problem is then to find $\mathbf{w}$ and b such that the constraints:

$$\mathbf{w}\ \Phi(x_i) + b \geq +1, \quad if\ \ y_i = +1 \tag{2.7}$$

$$\mathbf{w}\ \Phi(x_i) + b \leq -1, \quad if\ \ y_i = -1 \tag{2.8}$$

are satisfied [42]. These constraints can be combined to:

$$y_i(\mathbf{w}\ \Phi(x_i) + b) \geq 1 \quad \forall i \tag{2.9}$$

and the margin can be easily computed as the distance between the dotted lines:

$$M = \frac{|1-b|}{||\mathbf{w}||} - \frac{|-1-b|}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||} \tag{2.10}$$

where $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$.

Hence, the maximum margin can be constructed by solving the following optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \quad subject\ to\ \ y_i(\mathbf{w}\ \Phi(x_i) + b) \geq 1 \quad \forall i \tag{2.11}$$

Equation 2.11 can be written using the Lagrangian formulation:

$$\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha) \equiv \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i y_i(\mathbf{w}\ \Phi(x_i) + b) + \sum_{n=1}^{N} \alpha_i \tag{2.12}$$

where α_i ($\alpha_i \geq 0$) are the Lagrange multipliers for each constrain in Equation 2.11.

Based on the Karush-Kuhn-Tucker conditions, it can be shown that the optimal hyperplane is defined as a linear combination of the vectors in the training set [87]:

$$\mathbf{w} = \sum_{j=1}^{N} \alpha_j y_j \Phi(x_j) \tag{2.13}$$

under the condition:

$$\sum_{j=1}^{N} \alpha_j y_j = 0, \quad \alpha_j \geq 0 \tag{2.14}$$

The problem we introduced in Example 2.16 is linearly separable, but in real text classification problems, the data of documents in the feature space are seldom linearly separable. In this case, a non-linear mapping Φ is used to map the input space to a higher dimensional space. Such mappers are called *kernel functions* and allow to have a decision surface for non-linearly separable data.

$$\Phi(x_i)\Phi(x_j) = K(x_i, x_j) \tag{2.15}$$

Typical choice for kernels are [87, 81]:

- Linear Kernel: $K(x, y) =< x, y >$
- Polynomial Kernel: $K(x, y) = (< x, y >)^{\mathrm{d}}$
- Gaussian Kernel: $K(x, y) = exp(\frac{-||x-y||^2}{2\sigma^2})$

Above we discussed an example of SVM that is formulated for binary classification tasks. However, in the most practical text classification problems, the documents can be assigned to one or multiple categories from a category set. The way to overcome this issue is to combine several binary SVMs to create a single multi-class SVM. Many approaches (e.g. *one-against-one* and *one-against-all*) have been proposed, based on this idea and other voting strategies [87]. A comparison of these methods can be found in [46].

2.6.2 Text Clustering

As we mentioned at the beginning of the previous section, the problem we faced when trying to classify the data using the algorithms above was the lack of training data with the corresponding output (the class of each document). Hence, we decided to use clustering approaches to cluster the documents without relying on any data set. In the this section, we introduce clustering techniques that we used in this project.

2.6.2.1 Clustering Techniques

Opposite to classification techniques, clustering aims to discover unknown structures in a data set without additional knowledge. The task is to find objects that share certain similarities, and then group them into distinct clusters [88]. For example, we can consider documents containing keywords about engines and wheels

to be in category *cars*, while those containing keywords about telephones and computers in the category *electronics*, without having to refer to every object in those categories [79, 77].

Traditionally, clustering strategies are divided into various major models:

- **Hard clustering**: A document belongs to a cluster completely or not.
- **Soft clustering**: Each document have a probability of belonging to a cluster.
- **Hierarchical clustering**: A document belongs to a child cluster and also to the parent cluster (kind of tree).
- **Partitioning clustering**: Documents are grouped into clusters where every document is either hard clustered or has some documents in common (soft clustering).

Generally, clustering algorithms can be grouped based on the strategy used to cluster the data. The next discussion will show some clustering algorithms that we used in our project.

K-means

K-means clustering is an unsupervised hard clustering technique that tries to find a specific number of clusters k, represented by their centroids [70, 79]. This approach works as follow:

1. The user first choose the number k of the desired clusters.
2. K points are randomly created as initial centroids.
3. Assign the points to their closest centroid.
4. Update the centroid of each cluster based on the mean of the points in the cluster.
5. Go to step (3) if the centroids do not remain the same (convergence criterion).

These steps are illustrated in Figure 2.17, based on documents from three classes (square, circle, and triangle). In each of these figures, the centroid of each cluster is indicated by black stars. In the first iteration, the number of centroids is initialised ($k = 3$) and the documents are assigned to each of these centroids. When the groups are built, the centroids are updated (based on the mean of points) and the documents are assigned again to these centroids (this can be seen in Figure of iteration 2). These steps are then repeated until the algorithm terminates (when no more changes occur; see iteration N).

To assign a document to the closest centroid, a distance measure is used. This has to be relatively simple, since the positions of documents to each centroid are repeatedly calculated. There are many metrics that can be used for this purpose. For example, measures based on the *Minkowski metric* (also called L_q norm), denoted in Equation 2.16, can be used to calculate the distance d between two objects x and y having n feature [77].

$$d(x,y) = L_q(x,y) = \sqrt[q]{\sum_{i=1}^{n}(x_i - y_i)^q} \tag{2.16}$$

Another metric that is better suited for text documents [77, 11] is the *cosine metric* (Equation 2.17). It measures the similarity of two objects x and y by calculating the cosine of the angle between their feature vectors. Note that the output of Equation 2.17 will range between the values -1 and $+1$, where $+1$ denotes the highest similarity degree and -1 the lowest dissimilarity degree.

$$sim(x,y) = cos(x,y) = \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} * \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.17}$$

We mentioned before that the values of each centroid c_i (i refers to the $i^{th} cluster$) is randomly generated at the initial step, when the similarities of documents to the centroids is calculated. The documents are then assigned, and the new centroids are again calculated using Equation 2.18.

$$c_i = \frac{1}{m_i} \sum_{x \in cluster_i} x \tag{2.18}$$

where m_i is the number of documents in cluster i.

K-means is a simple, effective, and easy to implement clustering algorithm but has the drawback that the user has to predict the number of clusters K [79].

K-modes

The basic concept of k-means rests on the determination of similarities of documents in step 3, using similarity measures, and updating the centroids in step 4, using the mean function. But in some cases, the data which has to be clustered can be categorical, discrete, and unordered. For instance, Table 2.10 shows an example of attributes that describe films. The values of columns can be transformed into numerical values; the result will still be categorical values, which are not well suited to be used with distance measures discussed in the last section.

Film ID	Production Company	Main Actor	Producer
1	Walt Disney Pictures	Johnny Depp	Gore Verbinski
2	Imagine Entertainment	Johnny Depp	Rachel Talalay
3	Escape Artists	Nicolas Cage	Gore Verbinski
4	Saturn Films	Nicolas Cage	Marc Abraham
5	Imagine Entertainment	Russell Crowe	Brian Grazer

TABLE 2.10: Example of categorical data set

A modification of the standard k-means algorithm was proposed in [28]. This extension is called k-modes. It tries to replace the similarity distance with dissimilarity measure, and calculates the centroids using the mode function instead of the mean.

For instance, assume we have two categorical objects x and y that have n attributes. The distance function in k-modes is defined as [27]:

$$dism(x,y) = \sum_{i=1}^{n} \Phi(x_i, y_i) \tag{2.19}$$

where

$$\Phi(x_i, y_i) = \begin{cases} 0, & \text{if } x_i = y_i \\ 1, & \text{if } x_i \neq y_i \end{cases} \tag{2.20}$$

The function in Equation 2.19 is referred to as *simple matching dissimilarity measure* or *hemming distance*, and means that the larger the number of mismatches of categorical values between the two variables, the more dissimilar the objects. Note that in case of a mixed data set, where the data has numerical and categorical attributes, both algorithms can be combined to build the clusters. This approach is called *K-prototype* [78].

2.6.2.2 Evaluation Metrics

The objective when evaluating classification algorithms is to assess if the classifier would achieve more correct than incorrect predictions. This can be done using several metrics that compare the algorithm-assigned class of a document and the real category assigned by a human expert. To illustrate, assume that Ψ is a binary function that classify documents based on their content. A single document can belong to either a positive or a negative class. The output of the function Ψ will be compared to the decision of a humane expert who has labeled the documents manually. The comparison of the decisions can be summarized as follow [45]:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (FN)

TABLE 2.11: Confusion matrix

- **True positive**: The outcome of documents that are predicted as positive, and their actual label (the decision of the expert) is also positive.
- **False positive**: The outcome of documents that are actually negative, but the function Ψ predicted them as positive.
- **True negative**: The outcome of documents where the function Ψ and the humane expert agree that they are negative.
- **False negative**: The outcome of positive documents that are predicted as negative.

Now, if we want to measure the *precision* of the function Ψ (the estimated probability that documents which are predicted as positive are correctly classified), we can use Equation 2.21:

$$precision = \frac{\#TP}{\#TP + \#FP} \tag{2.21}$$

where $\#TP$ denotes the true positive rate, and $\#FP$ the false positive rate [45, 56].

Another measure that can be used to express the estimated probability of positive documents that are correctly predicted is the *recall* measure [45, 56]. This is defined in Equation 2.22:

$$recall = \frac{\#TP}{\#TP + \#FN} \tag{2.22}$$

Generally, the usage of these measures depends on the objectives. Precision is good as evaluator when the costs of FP is high. For example, FPs in an email spam classifier referring to emails that are ham[7], but has been classified as spam. In this case, the user could lose important emails if the precision is not high. Recall, on the other hand, should be used when the cost of FNs is high. For example, a model that detects the human immunodeficiency virus (HIV) might cause extremely high costs if it tells a patient (actually infected) that he is not sick.

There are tasks where it is important to have high precision and recall (e.g. our case with text classification). Seeking a balance between these two measures is a difficult task. If you improve one of them, you can degrade the other. A measure that can be used to find the harmonic mean of the precision and recall is the *F-measure* (also called F_1-Score) [56, 58]:

$$F_1 = 2 \times \frac{precision * recall}{precision + recall} \tag{2.23}$$

The F-measure reaches its best value at one, which means that the precision and recall are best, and its worst value at 0.

Until now, we have discussed measures to evaluate classification models as described in section 2.6.1.3. But if we deal with clustering problems where the expert-assignment of documents does not exist, these methods can not be used. In general, evaluating the quality of clustering algorithms is not a trivial task. In most applications, expert judgments are still required. In the literature [59, 18, 65], evaluation measure methods are divided into two categories: internal and external measure.

- **Internal measures**: In this category, the clustering result provided by an algorithm is evaluated based on the same information of data that was clustered itself. Here, the algorithm is given a quality score based on the cohesiveness of the clusters (the overall similarity of objects inside each cluster), and how well they are separated (dissimilarity to objects from other clusters). E.g., for the algorithms discussed in section 2.6.2.1, we can compute the similarity between each data point and its centroid. Additionally, we measure the distance between the centroids to see if the clusters are well separated. If we want to

[7] **Ham is an email that is not spam. https://wiki.apache.org/spamassassin/Ham**.

combine both measures, we can compute the so-called *silhouette coefficient* [67]. This is defined as:

$$s(i) = \frac{b(i) - a(i)}{max(b(i), a(i))} \tag{2.24}$$

where $b(i)$ refers to the smallest average dissimilarity between a data point i (e.g. centroid of a cluster) and all points in any other cluster, of which i is not a member (centroid of other clusters), and $a(i)$ is the average dissimilarity between i and the other data points in the same cluster. The silhouette coefficient varies between $+1$ and -1, where the best value is 1 and the worst value is -1. A value close to 0 indicates overlapping clusters, and negative values indicate that the data points have been assigned to the wrong cluster.

- **External measures**: In this category, the quality of clusters is evaluated using data that was not used for the clustering. This data is similar to the data that has been clustered, but is manually labeled by human experts.
  Note that the same evaluation approaches discussed earlier are used here.

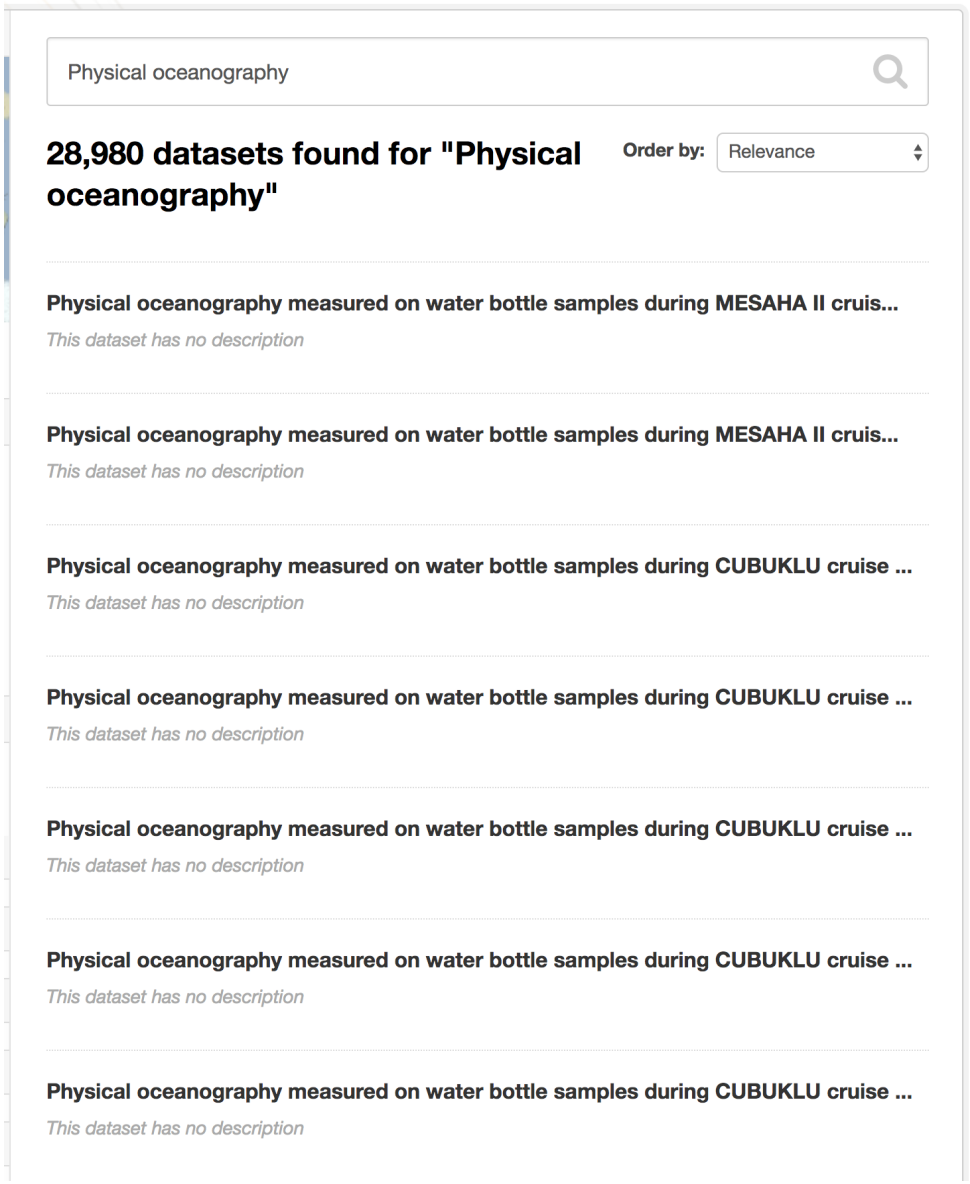


FIGURE 2.15: A screen capture of the result returned when searching for the search term **physical oceanography**. The user needs to visit each row to discover the difference between every document, as the titles are equal

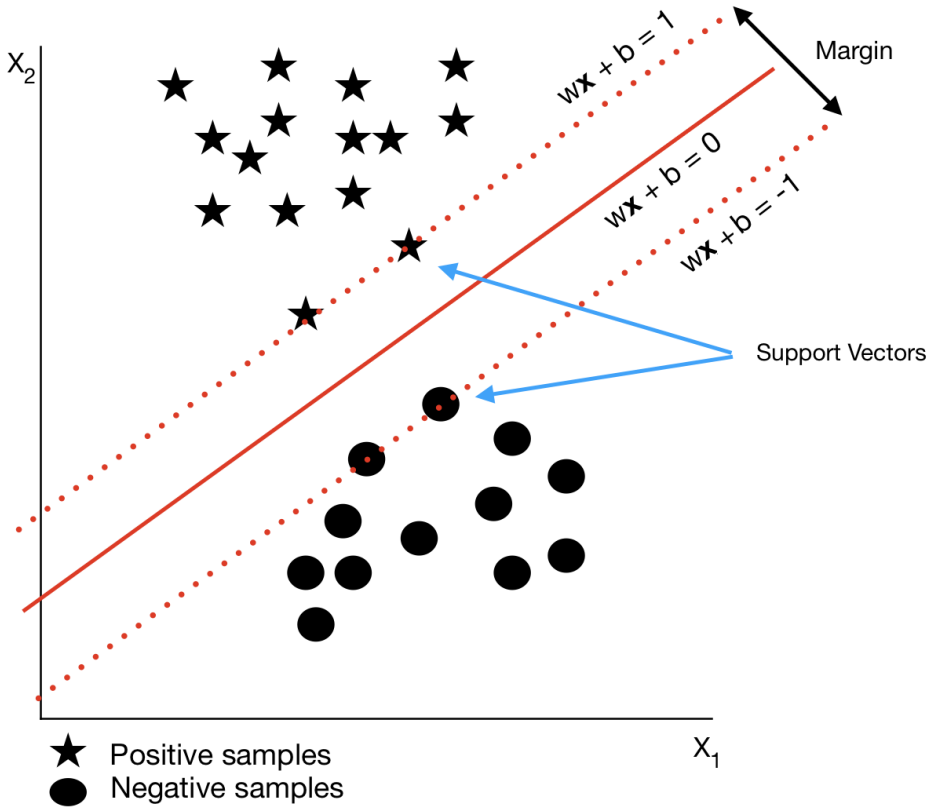
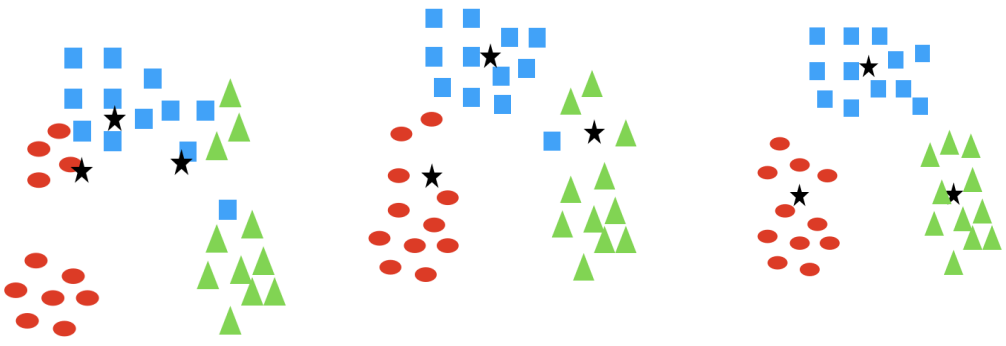


FIGURE 2.16: Support Vector Machines. A hyperplane separating two classes with the maximum margin



(A) Iteration 1 (B) Iteration 2 (C) Iteration N

FIGURE 2.17: *K*-means example.

Chapter 3

Implementation

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong." - Albert Einstein

3.1 Introduction

In this chapter, we will describe the experimental setup. First, we will give a description of the design of the system framework. Then we will describe the data set we used and provide some statistics about it. We will also mention which choices were made regarding the attributes and data selected. Furthermore, the pre-processing and transformation of this data will be explained. In addition, we describe the technologies that were used to build the categorisation models. We continue with a brief introduction on search machines and describe the schema used to index the data. We will end this chapter with a detailed comparison between various layouts that implement the navigation approaches discussed in the previous chapter.

3.2 Overview

Fig. 3.1 shows the designed workflow of the system. Three main components are managing the communication, the **Document Manager**, the **Search Manager** and the **Index Manager**. The system is designed to offer a flexibility when integrating a new language, implementing different approaches to feature extraction or using new classifiers.

- **Document Manager**: This component is responsible for document clustering, it has many modules for reading and extracting important data, detect the language and classify it.
- **Document Indexer**: This component is responsible for data persistence, it loads documents into the file system and indexes it to the search engine.
- **Search Manager**: This component is managing the search flow, takes keywords from the user, prepare the queries, searches for documents based on these queries, and returns the result.

3.3 Data Set

As mentioned earlier, the data set that should be categorised into groups is a dump of B2FIND service. It contains entries with various columns of metadata, exported

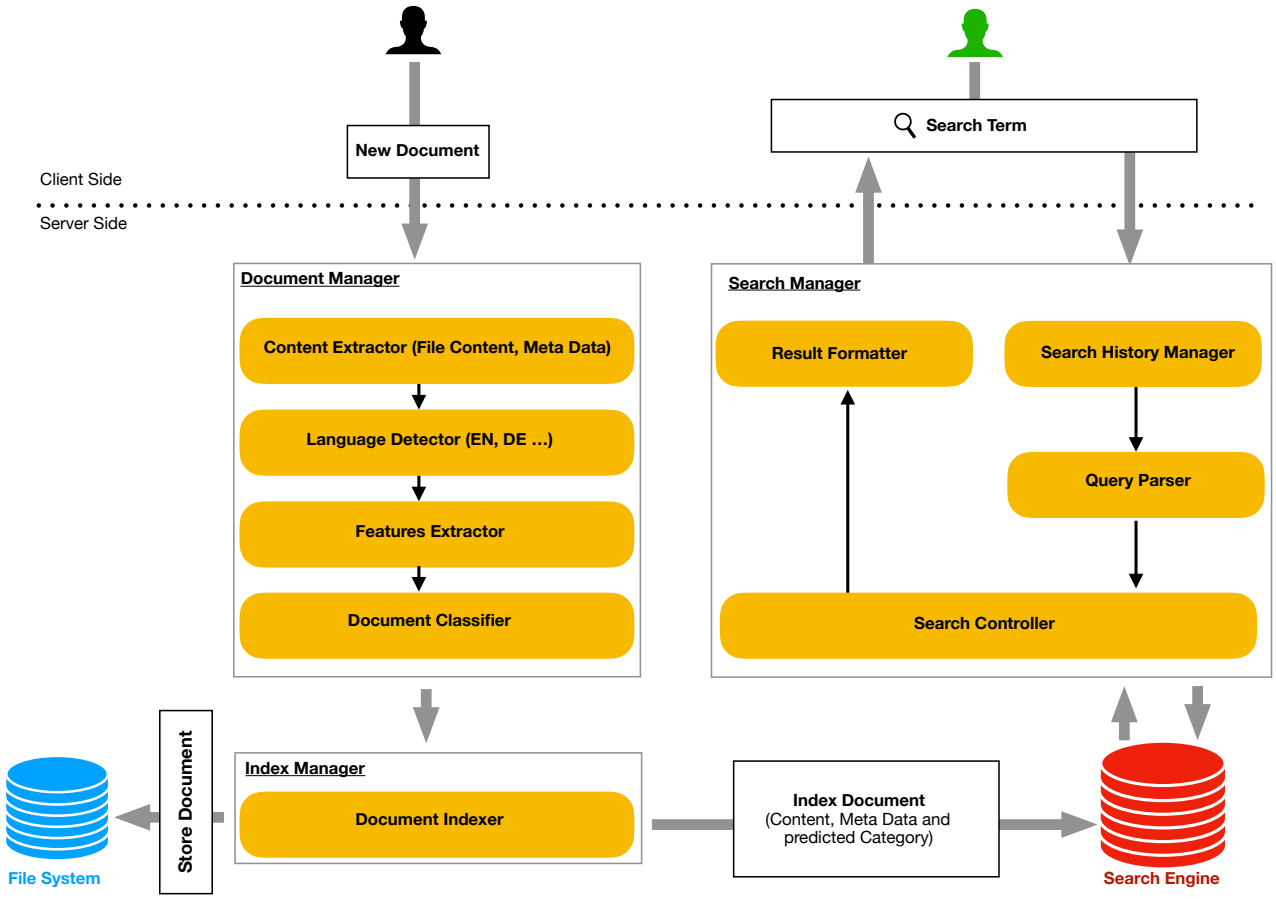


FIGURE 3.1: Overview of the architecture for the solution proposed

in CSV[1] format. A special column of this data is the **EXTRAS** column, it consists of keys and their values separated with three dollar $$$ signs that had to be parsed in order to make them suitable for our purpose. Generally, the content of the data is not consistent, many values exist only with some specific type of documents or publishers. Table 3.1 shows the attributes of data sets and their occurrences.

The element **ID** is the primary key of the set and represents the total number of entries. The field **STATE** is a helper field which is used to show or disable an entry, the rest of columns were explained in Table 2.2.

For the clustering task, we used the **TITLE** field, combined with **TAGS**, **GROUPS** and **DISCIPLINE** columns, and then evaluated supervised and unsupervised approaches. For the first approach, we had to have a training set containing documents and its classes. The reason why we decided to use the Wikipedia's category system[2] is that this corpus consists of more than 4.4 million articles with categories and their subcategories[3].

For the unsupervised approach, we used the B2FIND dump itself to extract the features and cluster the documents. The modules of **Document Manager** are described in the next sections.

[1] **Comma-separated values**.
[2] **https://en.wikipedia.org/wiki/Portal:Contents/Categories**.
[3] **https://corpus.byu.edu/wiki/**.

Attribute	Count
ID	934524
STATE	737912
TITLE	737915
AUTHOR	616045
URL	468337
NOTES	552360
TAGS	258225
GROUP NAME	625564
GROUP TITLE	625563
GROUP DESCRIPTION	625563
GROUP IMG URL	625563
EXTRAS	625563

TABLE 3.1: The occurrence of each attribute in the data set

3.4 Document Manager

This component has two main responsibilities: The first one is to process the copra by extracting and preparing their data, the second is to cluster or classify the documents.

3.4.1 Pre-processing

We have to extract two key-value pairs from the field **EXTRAS**; the LANGUAGE and DISCIPLINE. The reason for extracting the first one is to separate these entries based on their values in order to build multiple classifiers for each specific language. The second field is parsed for classification tasks. The language field does not exist in all entries, therefore we use a language detector in some cases. Figure 3.2 shows the occurrence of documents by their language. English is dominant, followed by Italian, Dutch, German and French.

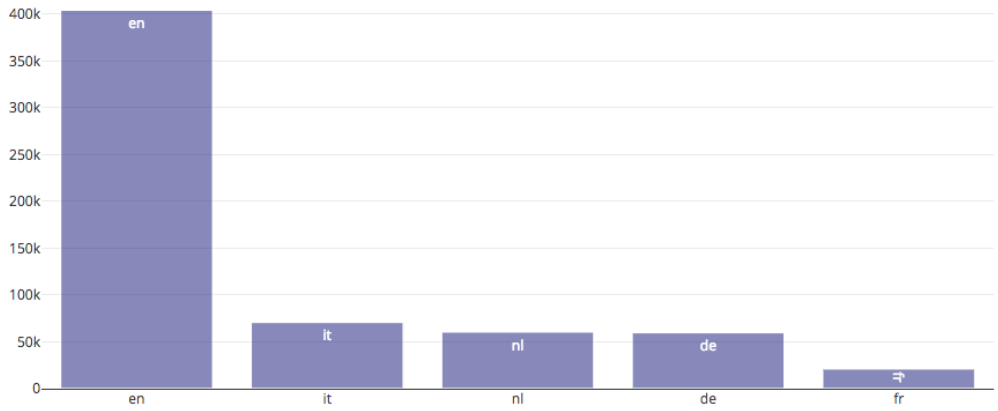


FIGURE 3.2: Top 5 languages in B2FIND

3.4.2 Feature Extraction

Four functions are part of this module: The first removes the stop words, the second stemms the textual content, the third generates the *N*-grams, and the last builds the matrix containing the features and their importance.

Stop-word Removal

We use the Python library Natural Language Toolkit (NLTK) to get the list (blacklist) of words to remove. We have to extend this list because of two issues we discovered. The first one is because some removed words are useful for our project, ergo we exclude them from the blacklist. And the second problem is that some keywords are not useful as features and could not be removed using feature whitening (as they are part of many duplicate documents). For this reason, we extend the blacklist to encompass more of such words.

Stemming

Stemming of textual data is conducted using the Porta-English stemmer from the Python library NLTK. This library has many functions for stemming words from different languages.

N-grams

We experimented with different lengths of n-grams (1, 2-, and 3-grams) and decided finally to use 1-gram. We used the `ngrams` function from NLTK.

TF-IDF Vectorization

Before we start categorizing the data, we use the `CountVectorizer` class to count the number of occurrences of every word, and the `TfidfTransformer` class to normalizes these counts and weight them. The result is the matrix containing words with its importance. Both classes are part of the Scikit-learn package for feature extraction.

3.4.3 Building up the Data Models

As we mentioned in Section 3.3, the first attempt to categorise the data was using supervised approaches. This category of algorithms needs a pre-defined data set to learn from. Hence, we used the categories of Wikipedia's category structure[4] as labels and their articles as documents in order to generate the features [89]. Note that there is a Python API[5] that can be used to get the Wikipedia pages of a specific category (see example in Listing 3.2). The training of the multinomial NB model discussed in Section 2.6.1.3, and the SVM discussed in Section 2.6.1.3 is conducted using the `MultinomialNB`[6] and `svm`[7] modules from the *scikit-learn* library. For the unsupervised approach, there was no need for a pre-defined data set, the B2FIND dump is used to generate the features and calculate the similarities of documents.

[4] **https://en.wikipedia.org/wiki/Portal:Contents/Categories**.
[5] **https://pypi.org/project/Wikipedia-API/**.
[6] **https://scikit-learn.org/stable/modules/naive_bayes.html**.
[7] **https://scikit-learn.org/stable/modules/svm.html**.

We used the prediction output (cluster ID), generated by the *k*-means[8] algorithm based on the **TITLE** and **TAGS** fields of the documents, as input for the *k*-modes[9], combined with the **GROUPS** and **DISCIPLINE** fields. More about the result of each setup is discussed in Chapter 4.

```
import wikipediaapi

wiki =wikipediaapi.Wikipedia(
    language= 'en',
    extract_format= wikipediaapi.ExtractFormat.WIKI
  )
wiki.page("Category:Physics")
```

TABLE 3.2: Code example to get pages of category physics from Wikipedia

3.5 Index Manager

This module handles the communication between the file system, search engine and document manager. It stores the clustering result in a file system and index the documents with their cluster IDs into the search engine.

3.5.1 Search Engine: Elasticsearch

Elasticsearch (ES) is a distributed search and analytics engine based on Lucene[10]. It allows to index, search, and analyze big volumes of data quickly and in near real time [41]. An instance of ES is called *node* and can form a cluster with other nodes.

In ES there are three concepts to follow in order to store and search for documents. These are *Indexes*, *Types*, and *Documents*. An index is a collection of containers that have documents sharing the same characteristics. For instance, one can have an index for customers, another index for products, and yet another index for transaction. Each container is a *type* and contains one or more *documents*. An index can potentially have more data than the capacity of the system. For example, an index of more than 2TB of documents may not fit the hardware capacity of the system, therefore an index can be divided into containers called *shards*.

Shards are distributed along several nodes and are responsible for searching, storing, and retrieving their own data [41]. Figure 3.3 shows an example of an ES cluster with two nodes having four indexes divided into four shards. One of these indexes is divided into three shards with one living in node 1 and the rest of shards are in node 2.

A document in ES is a unit of information converted into JSON and indexed with a unique identifier **_id**. ES automatically adds other metadata to this document such as the type, **_index**, where it lives and its type, **_type**.

To start storing documents in ES, it is required to create an index and set its configuration, as well as to define a document type with its fields. ES provides many

[8] **https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html**.
[9] **https://pypi.org/project/kmodes/**.
[10] **https://www.elastic.co/**.

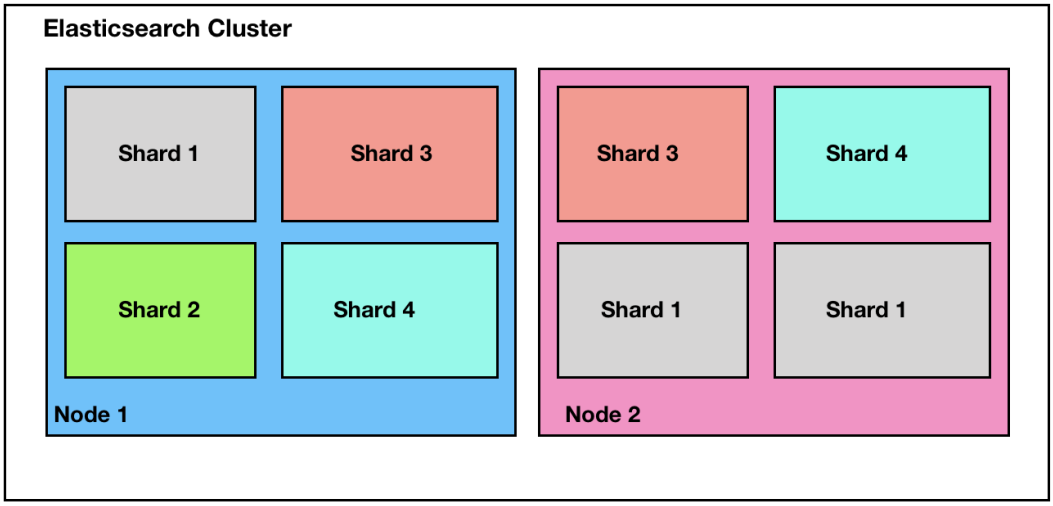


FIGURE 3.3: Example of an Elasticsearch cluster

parameters to set the number of shards, choose the full text searchable fields, and define fields that are stored but not searchable [5].

For this project, we used Elasticsearch 6.1[11] with two nodes deployed on two virtual machines[12]. Appendix B shows the mapping of the index used in this project. We have a *type* defined as document with various fields and their data type (keyword, text, object, and date). The field *cluster* is defined as an object list, because a document can have various cluster IDs depending on the cluster size selected. Next we explain some attributes of this configuration.

- **Settings**: An index has to have specific settings associated with it. These are defined in the body of definition using the parameter settings. For example, we can define the number of shards of the index, the number of replicas ot the shards, the maximum allowed difference between min_gram and max_gram when using ngrams.
- **Index**: The index attribute defines how the string will be indexed. Three values can be set for this field, "not_analyzed", "analyzed" (default), and "no". The first value is used for fields that are searchable but have to be indexed exactly as specified. The second value is for fields that are searchable and need to be preprocessed. For example, break the string to keywords, convert the keywords to lowercase. The last value is for fields that are stored but not searchable. For instance, an image link.

To create an index, a HTTP[13] request of type **PUT** has to be sent to the node address, followed by the name of indexes including its configuration in the body. In the same way, one can create documents by sending the same type of request to the node in the cluster with the URL **/{index}/{type}/{id}**, where **{id}** denotes the id that the document should be stored with.

[11] **https://www.elastic.co/products/elasticsearch**.
[12] **8GB RAM machines wtih Ubuntu 17**.
[13] **Hypertext Transfer Protocol - https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol\#Request_methods**.

3.6 Search Manager

The search manager contains four modules that handle the search queries and are responsible for a specific task:

- **Search History Manager**: As we mentioned earlier, we aim to offer the possibility to the user to navigate from cluster to sub-clusters in order to find the right documents. This can be done by memorising the previous terms and the clusters visited. This task is managed by the search history manager.
- **Query Parser**: This module takes the keywords of the user and generates results based on the queries that are compatible with the search engine.
- **Search Controller**: The search controller handles the communication between the query parser and Elasticsearch, it uses the elasticseach python client in its core to send HTTP requests.
- **Result Formatter**: This component converts the documents from raw text to the format used by the client. In our example, it generates the keywords and its importance for the word cloud chart.

3.7 Server

We handle the communication between all services shown in the last sections using a RESTful API. This is an application program interface that uses HTTP requests to communicate with many services to GET, PUT, POST, and DELETE data. The communication takes place inside a Flask[14] server through various endpoints called *routers*. Flask is a microframework written in Python and based on a Web Server Gateway Interface (WSGI) that defines many specifications for universal interface between web applications and web servers [19]. Figure 3.4 shows an example of an endpoint defined by Flask to return top documents based on a search query.

3.8 Front-End

The front-end is built in HTML based on Twitter Bootstrap[15] 4.0. This is an open-source front-end framework that offers HTML- and CSS-based design templates in order to develop web applications in a faster and easier way. The advantage of using Twitter Bootstrap is that it offers a responsive web layout that can adjusts dynamically, depending on the characteristics of the device used (desktops, mobile phones, tables) [73].
Each element of the interface triggers a function that sends an AJAX request to an end point of Flask. AJAX stands for Asynchronous JavaScript and XML, and is a set of web development techniques used for the development of interactive web applications. It allows, for example, web pages to change content without reloading the page itself [15, 84]. Figure 3.5 shows a code snippet of a Javascript function that sends an AJAX request to the *"/data"* endpoint. This function returns a promise that displays the result in the form of word cloud.

[14] **http://flask.pocoo.org/**.
[15] **https://getbootstrap.com/**.

```python
@app.route('/top_items', methods=['GET'])
def top_items():
    """
    Show top items of each cluster based on given query and filters
    :return: List of documents

    """
    if request.args.get("query", None):
        query = QueryParser.top_items_query(request.args.get('query'),
                                            request.args.get('filter').split(","),
                                            request.args.get('main_filter').split(","))

        docs = SearchController.fetch_top_items(query)
    else:
        docs = SearchController.top_products_by_frq()

    res = ResultFormatter.top_items_format(docs)

    response = app.response_class(
        response=json.dumps(res),
        status=200,
        mimetype='application/json'
    )
    return response
```

FIGURE 3.4: Example of an endpoint defined by Flask that returns top documents.

```javascript
let charsController = function (size, filter, main_filter) {
    loader.show();
    result_area.hide();
    jQuery.ajax('/data',
        {
            data: {
                "size": size,
                "filter": filter,
                "main_filter": main_filter,
                "query": search_field.val(),
                "delta_cloud": hidden_dcloud.val()
            },
            dataType: 'json',
            success: function (result, status, xhr) {
                loader.hide();
                freq_keys.val(result.freq_keys.toString());
                buildCharts(result)
            },
            error: function (jqXhr, textStatus, errorMessage) { // error callback
                console.log(errorMessage)
            }
        });
};
```

FIGURE 3.5: Example of a Javascript function that sends an HTTP request to a flask endpoint.

3.9 Navigation Experiments

To evaluate the accessibility and user friendliness of the project, we implemented different layouts containing various components, based on the approaches discussed in section 2.4.1. We assign each layout an identification number for reference in our discussion. Figures 3.7, 3.6, and 3.8 show screen captures of these layouts with their identification numbers on the bottom side of the page. The pages can be reached using the paths shown in Table 3.3 after starting the flask server[16]. Next, we introduce

[16] **How to configure and start fask: http://flask.pocoo.org/docs/1.0/quickstart/**.

each of these layouts and their navigation experience.

Layout	Link
1	/v1/index
2	/v2/index
3	/

TABLE 3.3: The path to each layout

Layout 1

In this layout, a user is able to select the number of clusters to start with. The result is a set of documents displayed in the form of colored markers in a scatter plot, each color represents documents inside a cluster. The user can disable the display of these documents by clicking on the legend of the cluster. Figure 3.9 shows a navigation example where a user searched for documents containing the keyword **education** and then filtered the result by only looking for documents from the group **CESSDA**. The result was a sub-set of documents clustered into 5 clusters (education, political education, sociology, and education). In this example, the *top documents* bar shows the result from eight clusters, and the number of documents remaining after searching and applying the filter. Note that clusters four, five, and seven were eliminated because they do not contain documents matching the search query. The user continued navigating (see Figure 3.10) by selecting documents belonging to the author, **University of London**. The result was one cluster with ten documents. Note that the title of the documents can be shown by hovering the marker on the plot, and opened by clicking on it.

Layout 2

The difference between layout 1 and layout 2 is the representation of the documents. As we mentioned before, our suggested solution displays the results in the form of a word cloud where each keyword shows its importance inside the cluster. Figure 3.11 is an example of a navigation experience of a user who was looking for documents about *Track-before-Detect Radar Systems*. The user started looking for documents about radars at the first step, the result was a set of documents grouped in 6 clusters (cluster 1, 2, 3, 6, and 8). The user filtered the result by choosing a group and an author, hence we see three documents remaining in Figure 3.11. In the last step, the user clicked on the word **agent** from the word cloud, thus the navigation system shows one document from cluster 2 (see Figure 3.12).

Figures 3.13 and 3.14 are used to demonstrate the difference between the representation in layout 1 and layout 2. The user aimed to navigate to documents that contain the keywords **weather** and **radars**. In the first layout, the drawback is that the scatter plot shows more than 1000 documents that match the query and displays only one difference between them (the publication date cited in the title).The user then has to hover over more documents to get more insight about the result, or use the publication date filter. The same result is illustrated in Figure 3.14, here the word cloud shows only two keywords, which means that based on the given query, the 1350 documents contain two significant differences: **data** and **month**. Each of these

FIGURE 3.6: Example of layout containing a scatter plot of the clusters in various colors. The right sidebar contains top products of each cluster and the number of documents inside it

words appears in many documents but not in all, the reason why the cloud shows only those and hides others, such as the word **measurement**, as it is found in all matched documents.

Layout 3

Layout 3 follows the **Hierarchical Navigation Approach** discussed in section 2.4.1. Figure 3.15 illustrates this appraoch using the navigation experience of a user who looked for documents matching the keyword **weather**. The result of his query was a set of documents clustered by disciplines. As a next step the user decided to search inside the sub-cluster **Environment Science** (see Figure 3.16) by looking for documents containing the words **weather** and **water**. The query matched one document and the result was clustered, this time by authors (see the breadcrumb in on the left sidebar).
The drawbacks of this navigation approach is that the pivot variable could have many unique values that would produce long-scroll pages and build clusters with small numbers of documents (in most cases one document per cluster as illustrated in Figure 3.17).

3.10 Navigation Testing

We carried out some tests with users to evaluate the usability and user friendliness of each layout presented. The users got test scenarios with different test tasks that needed to be completed.

- **Test users**: A user has to meet three criteria to be qualified to test the layouts. The first criterion is having some experiences with web technologies, such as

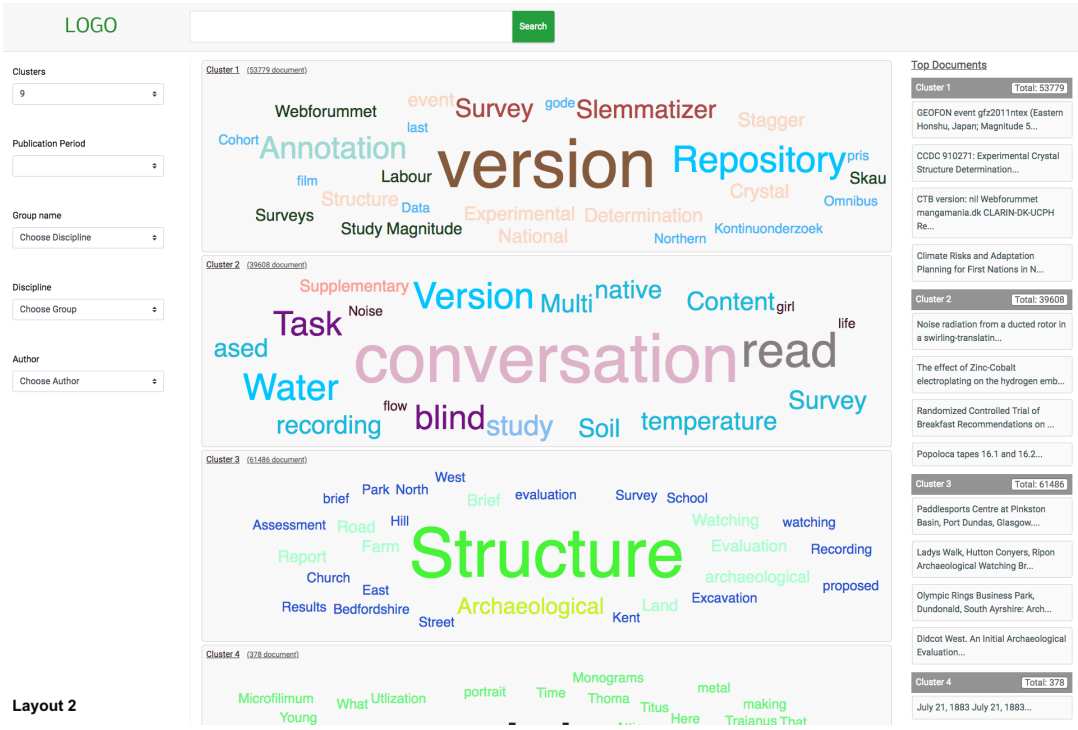


FIGURE 3.7: Example of layout containing the important keywords from each cluster. The left sidebar contains control elements to filter the results

working with a browser and knowing the functionality of HTML elements (inputs, checkboxes, buttons). The second criterion is that the user has to be a researcher or at least a student in order to understand some keywords of the word cloud. The last criterion is the user has to understand English, as the task scenarios are written in this language.

- **Task scenario**: A list of tasks that covers most elements in each layout. The tasks were developed in a way that even a user without knowledge about the disciplines or documents of a cluster could complete them (see Appendix E).

- **Test materials**: We used three test materials during the tests. The first one is a questionnaire that contains questions about the test user in order to understand his/her background. We used this document to record the output of the tests (Appendix C). The second is a post-task questionnaire for measuring the usability. We used a System Usability Scale[17] (SAS) (See Appendix D.1). The last is a document with which we logged some users' actions during the tests, and the time needed to complete a task. (Appendix F).

- **Test equipment**: We used the same laptop to carry out the tests to ensure that the layouts are rendering without any problems that could hinder the users completing the tasks. Each user did the tasks with his/her favorite web browser (Safari, Chrome, Firefox).

- **Test location**: We conducted the tests in the library of the University of Hamburg (UH) and the library of the Hamburg University of Technology (TUHH).

[17] **https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html**.

FIGURE 3.8: Example of layout containing the important keywords from each cluster. The right sidebar contains top products from the corpus

We chose these locations as they provided a high chance to meet test users that fulfill the criteria we mentioned before.

We started giving an introduction about the system, the reason why we do the tests, and showing some documents that the user has to navigate to. During the tests, we asked the users to verbalize their intentions in order to record them with his/her actions. The results of the tests are discussed in detail in Chapter 4.

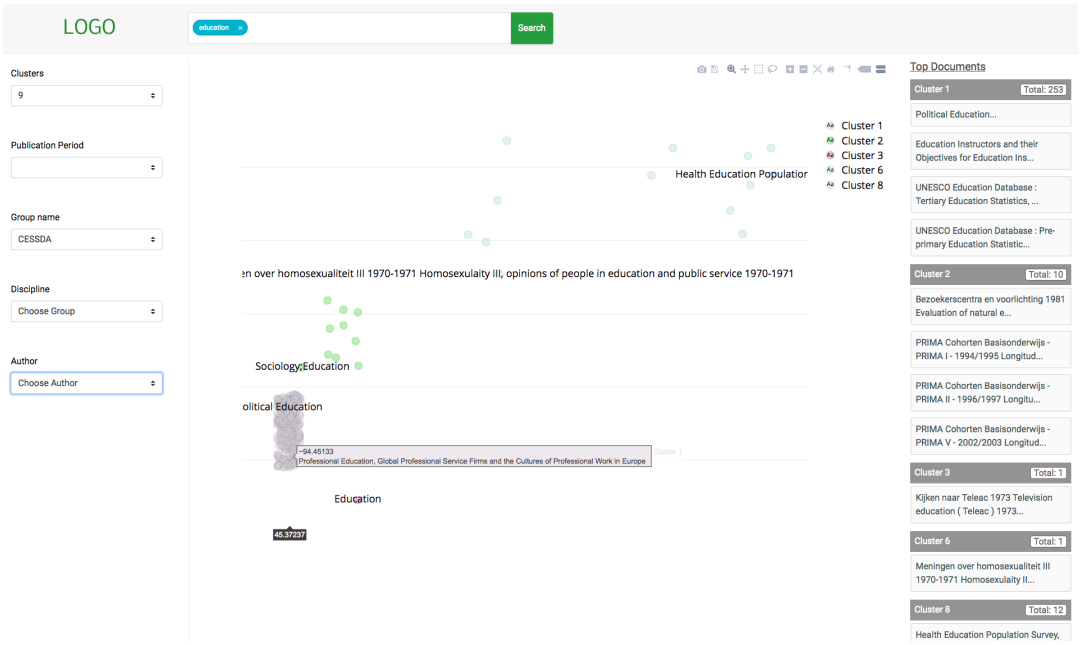


FIGURE 3.9: Example of result representation in layout 1. The user aims to navigate to documents about education from the group *CESSDA*

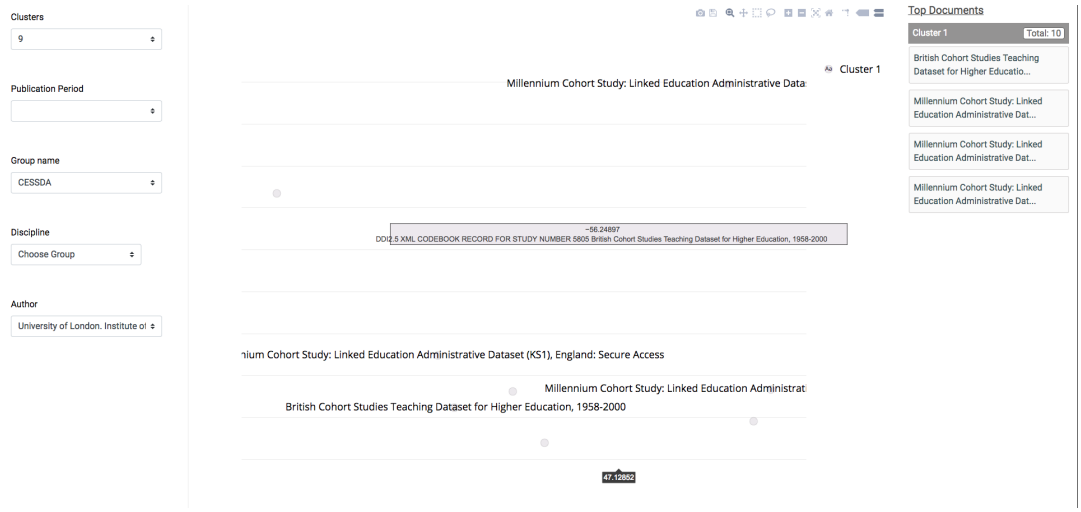


FIGURE 3.10: Example of a result representation in layout 1. The user aims to navigate to documents about *education* from the group *CESSDA* published by *University of London*

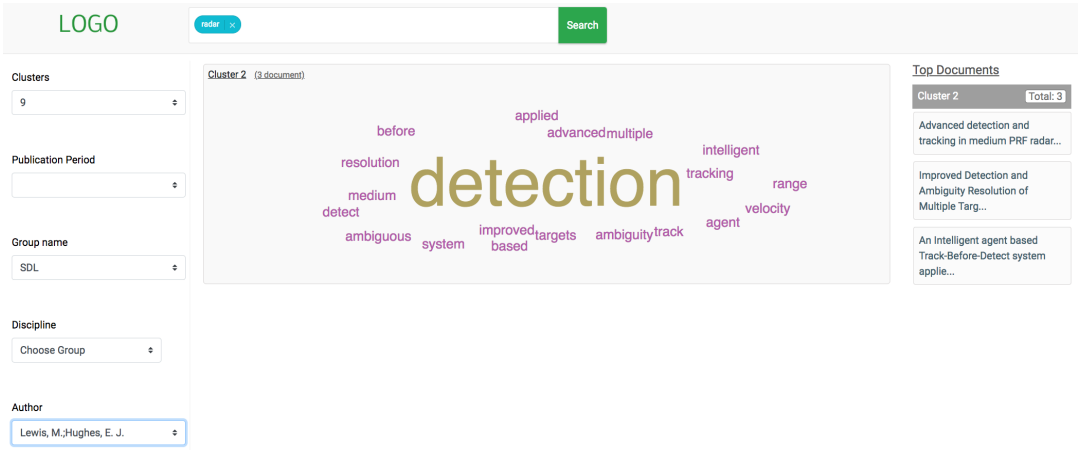


FIGURE 3.11: Example of a result representation in layout 2. The user aims to navigate to documents about *radars* from the group *SDL* published by *Lewis, M. and Hughes, E. J.*

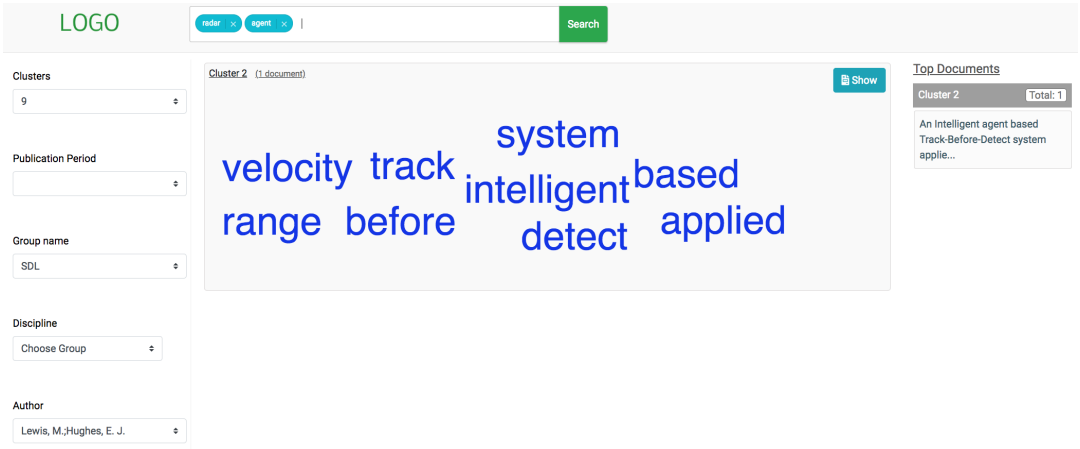


FIGURE 3.12: Example of a result representation in layout 2. The user continued his navigation from Figure 3.11 by looking for documents that match the word *agent*. The end result contains one document from cluster 2

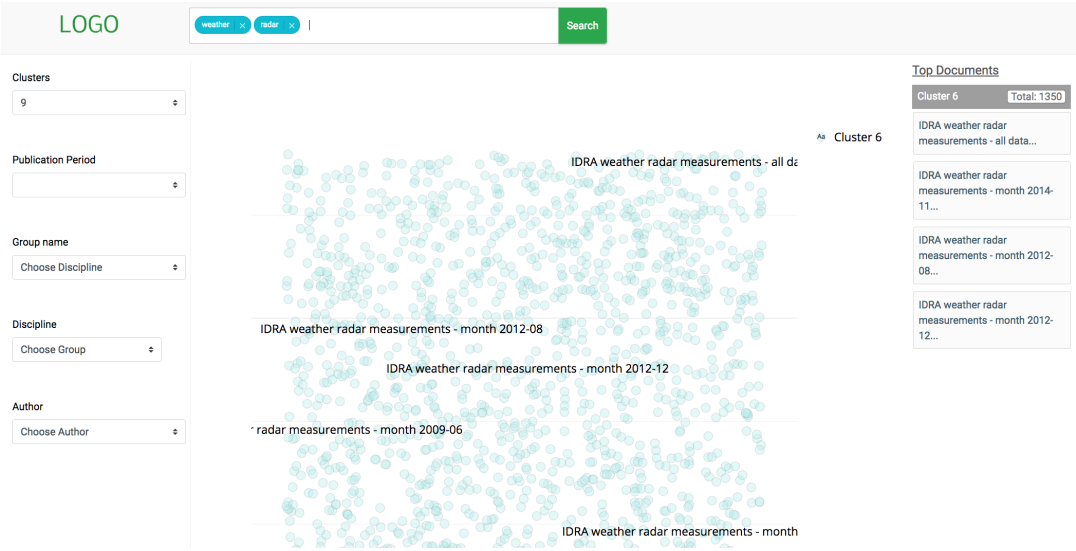


FIGURE 3.13: A navigation issue in layout 1. The user aims to find documents about *radars* and *weather*. The result contains 1350 different dataset. The user has in this case to hover many markers to see the content of document

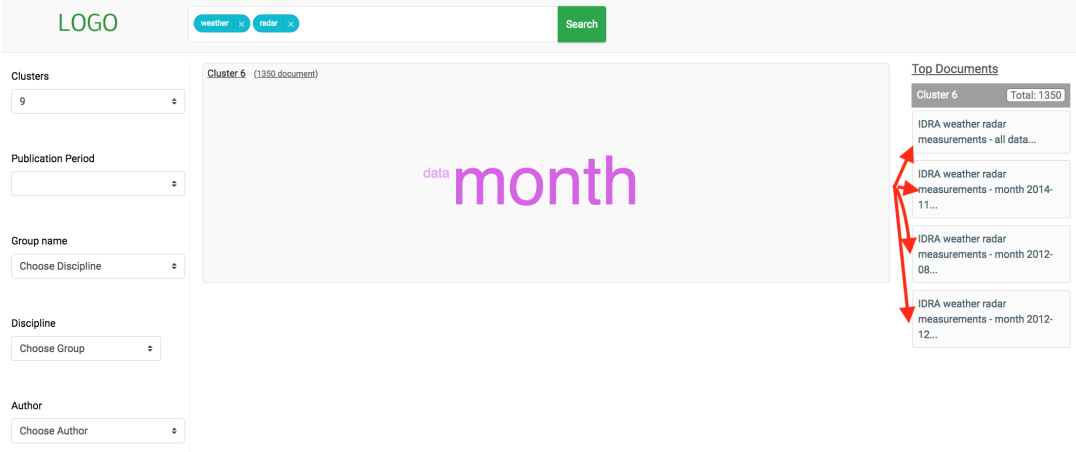


FIGURE 3.14: A navigation example that illustrates the same result as in Figure 3.13. The word cloud shows only two keywords which means that the 1350 documents contain the same content except two keywords that could help the user to navigate forward. If the word cloud would show the word *measurement* and the user would click on it, the result would be again 1350 documents

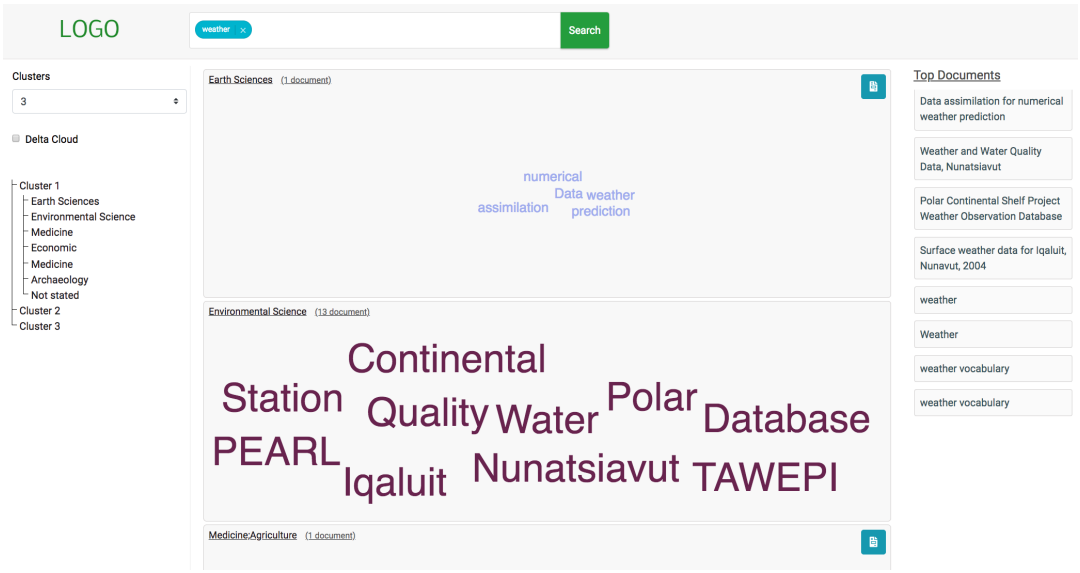


FIGURE 3.15: Example of a result representation in layout 3. The user aims to navigate to documents about *weather*. The result is grouped in disciplines and displayed in from of word clouds

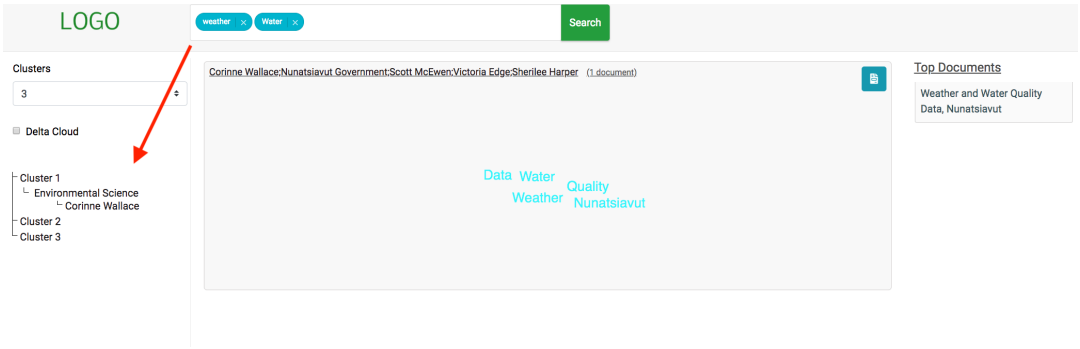


FIGURE 3.16: The user continued the navigation from Figure 3.15 by looking for documents that match the word *water*. The end result contains one document published by the group *Corine Wallace*.

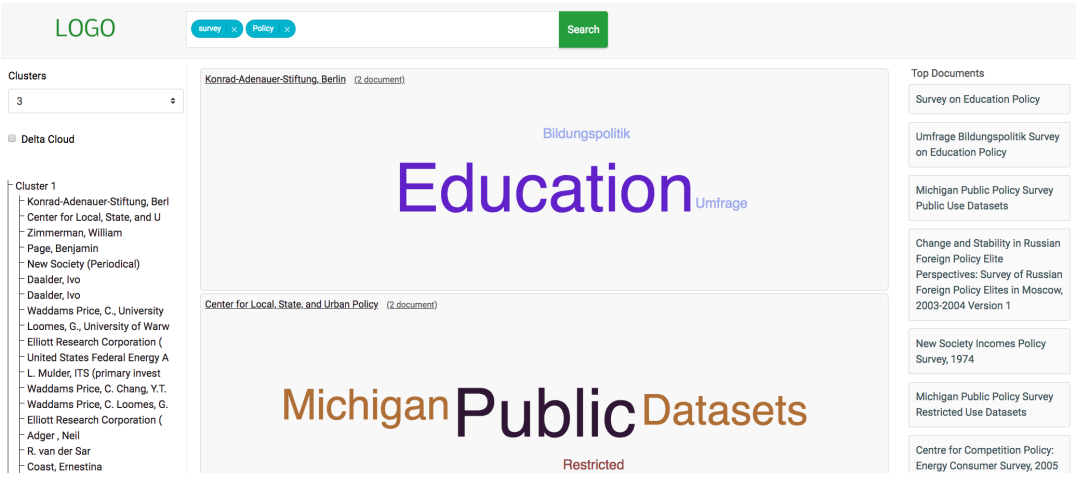


FIGURE 3.17: A navigation issue in layout 1. The result was categorised by authors which yielded to a long list of clusters containing few documents

Chapter 4

Experiments and Results

"To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science." - Albert Einstein

4.1 Introduction

In this chapter, we will present the results of our experiments[1]. We will first briefly discuss the performance of each selected model. Then we show the results of the experiment conducted with the test users. We continue with a review about the drawbacks of components in each layout, and end the chapter with a discussion about various improvement, that we will propose for each layout.

4.2 Categorization Results

Classification results

The initial idea was to use pages from Wikipedia and their parent category as training set to build a model (NB or SVM) that can be used to categorise the data. The model would calculate the probability that a document from B2FIND 's collection belong to a certain category. In this way, we could assign for each document the category with the highest probability. During the building process, we kept 80% of the data to train the models and used 20% for testing their quality. The result of these tests were actually satisfying (see Table 4.1), but when we applied the models on B2FIND 's collection, the prediction probabilities were very low. For this reason, we started validating the prediction results, and the documents content manually.

	precision	**recall**
NB	0.931	0.912
SVM	0.952	0.925

TABLE 4.1: Classification results of Wikipedia's data set

During the validation, we find three issues that impacted the prediction results: The first is that many documents contain in-descriptive titles. For instance, document number 1 in Table 4.2 has a kind of serial number or ID as title, which means that is very difficult even for a human being to predict its category (recall that we

[1] **The experiments were conducted using a virtual linux server with 15 GB Ram and 4vCPU**.

discuss supervised approaches in this section). The second is that many documents consist of a title with one keyword that does not appear in the vocabulary set which is built based on features using Wikipedia 's pages (e.g. document number 2 in Table 4.2). The solution to this problem is to extend the training set with more pages from other categories. But the issue with this solution is that the size of the article text in the English Wikipedia is approximately 14 GB[2] (compressed), which means that the data would not fit in the memory of our virtual server. The last is that in many titles the space between the keywords was removed, which led to meaningless terms (document number 3 in Table 4.2).

	Title
1	T3_06B_AO_CSPD
2	Trobriands_199_028
3	CoconutStoryWA30M

TABLE 4.2: Titles of documents from B2FIND

Finally, we decided to use clustering methods to avoid these issues. In doing so, we avoid using a data set that has to be extended every time documents with new keywords are added to the collection. In addition, in cases when the title contains in-descriptive content, we would use other attributes of the document to assign it to a the cluster.

Clustering results

As mentioned in Section 3.4.3 two methods will be combined to determine the clusters. The first method ($k-$means) is used to build clusters based on the title and tags of documents. Recall that these two fields are transformed into numerical data using the TF.IDF weighting. The intention is to convert the numerical data into categorical by determining the cluster ID of each document, and then providing the ID of the cluster, the discipline, and group name to k-modes. Table 4.3 shows the silhouette coefficient of the clustering results using 1-gram and 2-grams for different sizes k.

	2	**3**	**4**	**5**	**6**	**7**	**8**
1-gram	−0.023	0.114	0.186	0.122	0.244	0.249	0.110
2-grams	−0.063	−0.135	−0.962	−0.527	−0.481	−0.374	−0.413

	9	**10**	**11**	**12**	**13**	**14**	**15**
1-gram	0.087	0.064	0.092	0.051	0.038	0.032	0.017
2-grams	−0.431	−0.452	−0.491	−0.502	−0.484	−0.465	−0.486

TABLE 4.3: K-means silhouette coefficient by cluster size

By looking at the values we can determine that tokenizing the data using a 1-gram model performs better than 2-grams. The first observation is that the result based on the 2-grams model is getting worse (the negative values indicate that the documents have been assigned to the wrong cluster) when we enlarge the cluster

[2] **https://en.wikipedia.org/wiki/Wikipedia:Database_download**.

size k. This is due the facts that the documents-features matrix (recall the matrix presented in Table 2.8) become very sparse when using 2-grams based vocabulary, and that many documents contain only a title that is less descriptive, or consist of one keyword which appears only once. The second observation is that the best results we can get are using a 1-gram model with $k = 4$ or $k = 7$. Generally, all the values in Table 4.3 are close to 0 which means that regardless of the chosen parameter k[3], the clusters are still overlapped.

Table 4.5 displays the silhouette coefficient of the k-modes clustering results. In this table, the rows represents the cluster size of the k-means algorithm, and the columns the cluster size of the k-modes algorithm. Note that it is important to use the hamming distance when using the python `silhouette_score` evaluation function in order to generate coefficients of categorical data (see example in Listing 4.4).

```
from sklearn.metrics import silhouette_score

silhouette_avg =silhouette_score(data, cluster_labels, metric='hamming')
```

TABLE 4.4: Code example of the `silhouette_score` function in Python

The first observation is that all values (regardless of the k parameter of k-means) are positive which means that the documents are better clustered than the setup we had before (based on k-means and the title of documents). The next observation is that the values are no more very close to 0, but still far away from the best value $+1$. This means that the clusters are better built, but still slightly overlapped by some parameters k (for example by $k = 2$ or $k = 6$). The last observation is that using k-means with 7 clusters performs better in many k-modes cluster sizes than with 4 clusters. Recall that in all layouts, the user is able to choose the number of clusters he/she wants to display, therefore it is not required to select the best parameter here. We provide for a document a list that maps the cluster IDs and the parameters k of k-modes. Another notice is that in layout 2, the clusters are decoupled slightly to facilitate the navigation for the user. The objective is to avoid having many overlapped clusters that hinder the user reading the content of documents.

	2	**3**	**4**	**5**	**6**	**7**	**8**
k = 4 (k-means)	0.350	0.394	0.190	0.147	0.414	0.310	0.282
k = 7 (k-means)	0.297	0.315	0.320	0.403	0.258	0.360	0.355

	9	**10**	**11**	**12**	**13**	**14**	**15**
k = 4 (k-means)	0.349	0.433	0.402	0.397	0.439	0.491	0.440
k = 7 (k-means)	0.441	0.377	0.431	0.451	0.418	0.456	0.450

TABLE 4.5: K-modes silhouette coefficient by cluster size

[3] **As the clustering process took many hours and some times led to OUT_OF_MEMORY errors, we stopped evaluating numbers greater than 15.**.

4.3 Navigation Results

We conducted the experiment with 33 test users, whereby each was given the list of tasks that he/she had to complete and a layout that we chose randomly. We ensured at the end of the tests that each layout was tested by the same number of users from different subjects to avoid biased results. During the tests, some users could not complete some tasks, in which case we gave them help and logged these tasks as completed tasks with help. Figure 4.1 shows the distribution of tasks' completions on each layout. Most users who failed completing the tests worked with layout 3.

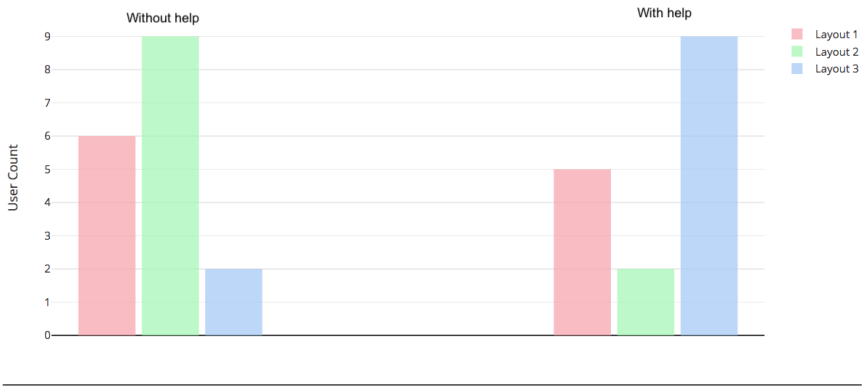


FIGURE 4.1: Number of users that completed the tasks with/without help on each layout

Layout 1

Using the first layout, five users of eleven are unable to complete the tasks without help; they failed either with task 1, 4, or 12 (see Figure 4.2).

- **Task 1**: The inability to complete this task is because most of users have not been working with scatter plots before, they did not realize that selecting a dataset can be done by clicking on a marker in the plot.
- **Task 4**: To select only documents from one cluster, the user needs to click on the name of other clusters to hide them (see Figure 4.3). At the end of the tests, only seven users could do this without any instructions.
- **Task 12**: Some users started hovering the markers to find datasets containing the date on the title. After a short search, they asked for help. This task can actually be solved by using the calender filter, but in same scenarios, the user needs to see the title of documents to decide.

Layout 2

Only two users could not finish the tasks given layout 2.

- **Task 8**: To clear a filter, a user has to choose the first option (e.g *select an author* by the author field). This is why most users who complete this task had to reload the page, or passed the task by chance.
- **Task 11**: Each cluster shows the size of the result it contains. Some users were confused with this representation because they thought that this size represents only the number of top documents inside the cluster.

Layout 3

Users of layout 3 had more difficulty in completing their tasks without help. The page was too long in some scenarios, hence some users could not see the cluster containing documents of the group, *Datacite* (task 3). Another issue was the name of the clusters, which were in some cases very long (see Figure 4.4).

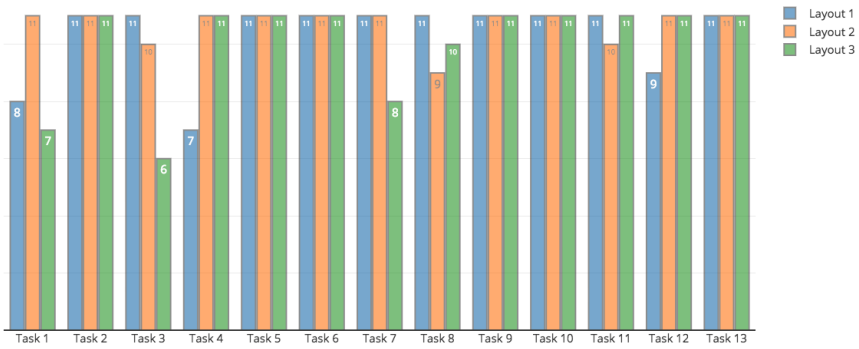


FIGURE 4.2: Success rate of tasks on each layout

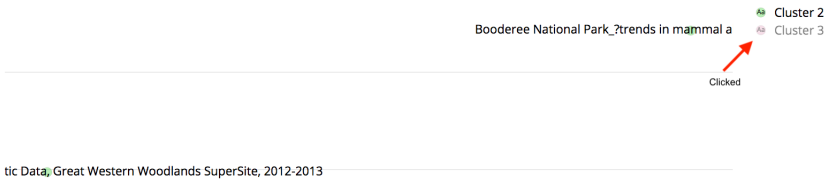


FIGURE 4.3: Hiding a cluster in layout 1

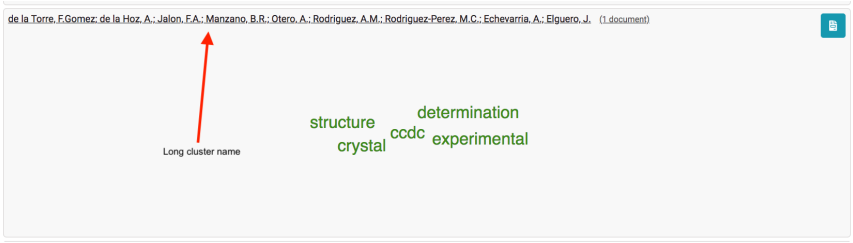


FIGURE 4.4: Issue of cluster title in layout 3

Concerning the time, users of layout 3 required the longest on average due to the fact that it displays long pages in some cases, and have components that confused the user. In layout 3, the user spent some time hovering over documents to get their name. Table 4.6 shows the result of a two-sample t-test that investigates whether the time means of the three layouts differ from one another. Since the p-value is much less than the test statistic for layout 1 and 2, the null hypothesis can not be rejected, which means that test users of layout 1 and 2 needed approximately the same time to complete their tasks. The test shows a significant difference of completion time on layout 1 and 3, and also 2 and 3 (see Turkey's test in Appendix G.1).

Layout	1		2		3	
	statistic	p-value	statistic	p-value	statistic	p-value
1	0.0	1.0	0.74	0.46	-2.99	0.007
2	0.74	0.46	0.0	1.0	-3.73	0.001
3	-2.99	0.0072	-3.73	0.001	0.0	1.0

TABLE 4.6: A two-sample t-test that investigates whether the means of time token in each layout differ from one another

In general, most test users agree, that layout 3 is not easy to use, complex, and that they needed support from a technical geek to be able to use it (see answers of SAS questions in Figure 4.8). The result from the group which tested layout 1 was unexpected (Figure 4.6); most of them agreed that layout 1 is easy to use and that most people would learn to use it very quickly. On the other hand, they consider the components of this layout badly integrated.
Layout 2 finished the test with better results than other layouts, most users agreed that it is easy to use and that the components were better integrated than in layout 1 (Figure 4.7).

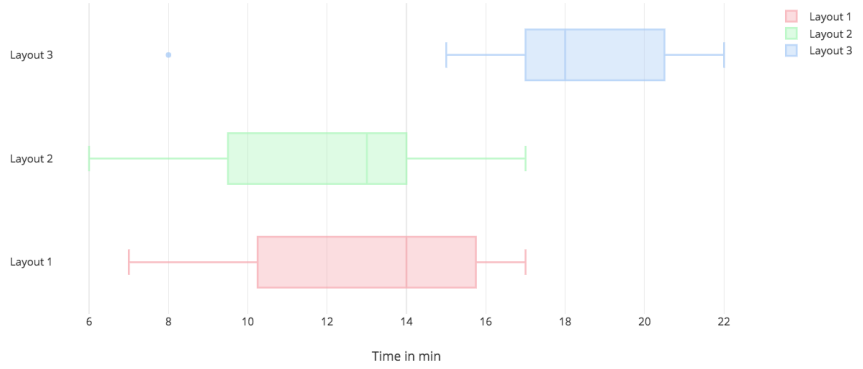


FIGURE 4.5: Boxplot of completion time on each layout

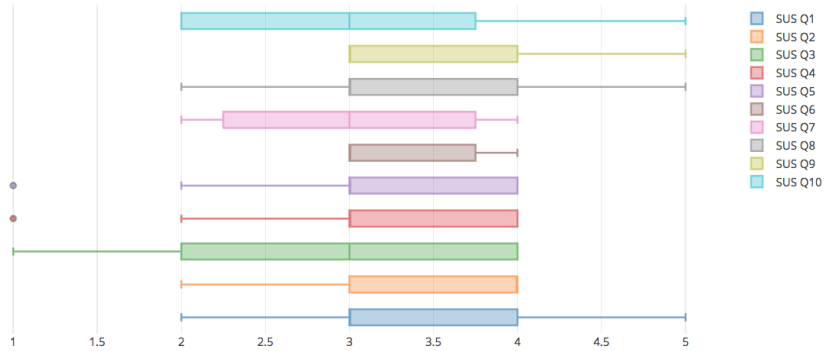


FIGURE 4.6: The mean scores of test users' responses on layout 1 in SAS. (Strongly agree = 1, strongly disagree = 5)

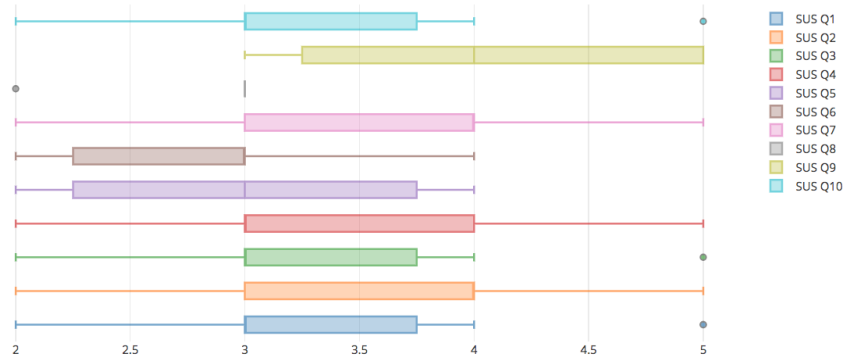


FIGURE 4.7: The mean scores of test users' responses on layout 2 in SAS. (Strongly agree = 1, strongly disagree = 5)

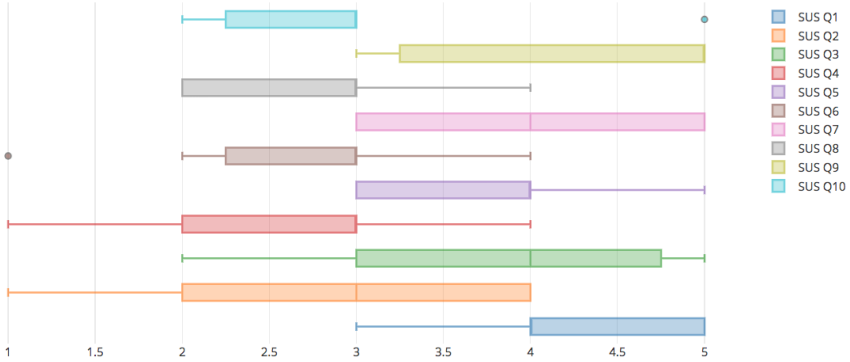


FIGURE 4.8: The mean scores of test users' responses on layout 3 in SAS. (Strongly agree = 1, strongly disagree = 5)

Chapter 5

Conclusion

"From the end spring new beginnings." - Pliny the Elder

5.1 Discussion

The methodology developed in this thesis enables to improve the productivity for users when searching or navigating large text collections. The idea is to use graphical alternatives to present the search results in category structure instead of the dominant list based approaches.

The thesis started with a theoretical study of the dominant approaches and discusses their components. The objective was to show the effort needed to precess each of these components. As next step, a navigation model was proposed that minimises this effort. The assumption is that if the effort of the individual components is reduced, the overall effort would be minimised. Thus, the whole navigation process was improved. In a third step, the thesis moved from theory to practical implementation. It discussed the implementation of three interfaces that assist the user in navigating text data. Each of these interfaces contains a set of components that proofs the concept presented before (the navigation model).
The thesis ends with a description of the evaluation method conducted with test users in order to test the quality of the interfaces and discusses the result of this evaluation: Generally, it cannot be said that one of these navigation assistances is superior tp the other. Each of them has its advantages and drawbacks. The graphical visualization in layout 1 enables the user to search for documents without scrolling the page too long, it also offers the possibility to categorise the documents into clusters with different colours, hide documents of a cluster from the result, and use the filters to exclude some documents. However, the user has to hover over the documents to see its content. In addition, the user has to think about strong keywords to navigate forward. Layout 2 improves these drawbacks by presenting the documents of the clusters in form of word clouds. This gives an idea about the documents inside the cluster about selecting important keywords to start with. However, the user has to scroll the page extensively if he increases the number of clusters at the beginning. In addition, keywords of the word cloud can be ambiguous if the categorisation algorithm does not separate the documents well. Layout 3 dispenses the filters on the red area and presents the documents in hierarchical clusters, based on a pivot variable. The drawback of this method is that in case the pivot variable contains many unique values, the page will be too long.

Comparing the three interfaces based on the answers of SAS questions, layout 1 and 2 perform better than layout 3. They are less complex and easy to use. However,

users need support during the first usage in order to understand the components. Hence, an automatic guide demonstrating the components of each interface step-by-step was added. It helps users to learn (see Figures 5.1 and 5.2).

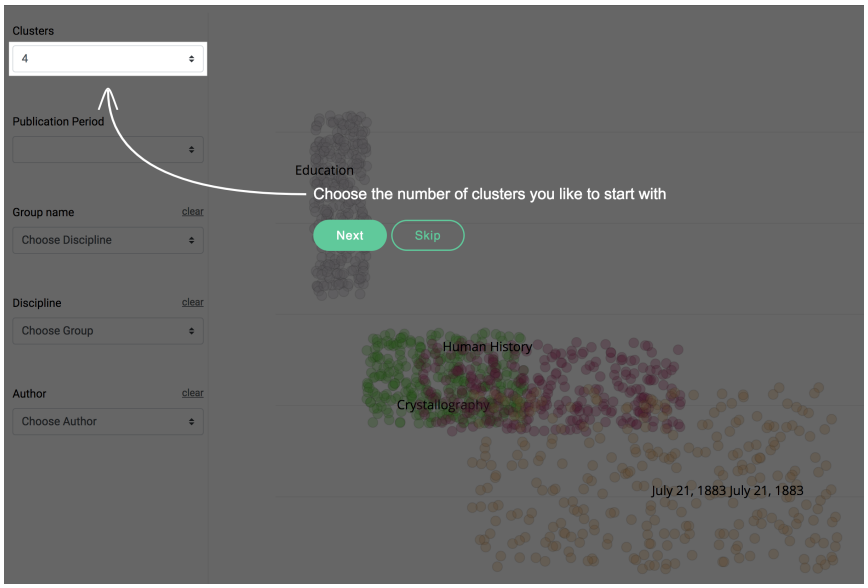


FIGURE 5.1: Example of an instruction of the guide in Layout 2

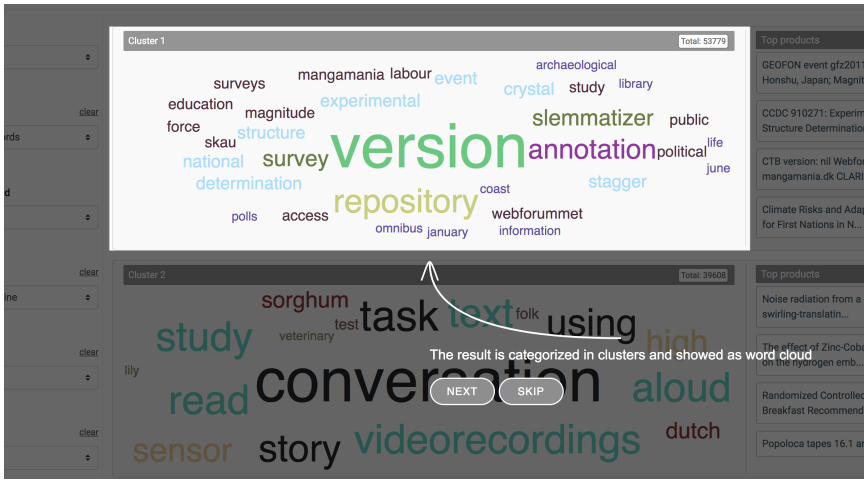


FIGURE 5.2: Example of an instruction of the guide in Layout 3

5.2 Future Work

Further work related to this thesis can go in two directions. The first direction is to use other visual presentations, for example 3D network graphs that show the clusters and their relationships [16]. The user could zoom inside the cluster and follow the links to other clusters until a target is found. The role of the links is to connect keywords that appear in many clusters and that are ambiguous. Another example is to find out about the impact of mixing the presentation methods depending on the content of the clusters.

The second direction is to take the user navigation history, the global search history and the user location into consideration, for example by using this data to apply learning-rank algorithms [21, 47] on the clustered documents to personalize the results.

Appendix A

Table of English Stopwords

Filter	Description
A	a, about, again, all, almost, also, although, always, among, an, and, another, any, are, as, at
B	be, because, been, before, being, between, both, but, by
C	can, could
D	did, do, does, done, due, during
E	each, either, enough, especially, etc
F	for, found, from, further
H	had, has, have, having, here, how, however
I	i, if, in, into, is, it, its, itself
J	just
K	kg, km
M	made, mainly, make, may, mg, might, ml, mm, most, mostly, must
N	nearly, neither, no, nor
O	obtained, of, often, on, our, overall
P	perhaps, pmid
Q	quite
R	rather, really, regarding
S	seem, seen, several, should, show, showed, shown, shows, significantly, since, so, some, such
T	than, that, the, their, theirs, them, then, there, therefore, these, they, this, those, through, thus, to
U	upon, use, used, using
V	various, very
W	was, we, were, what, when, which, while, with, within, without, would

TABLE A.1: Example of English stop words

Appendix B

Elasticsearch

```
PUT app
{
   "settings" : {
      "number_of_shards" : 2
   },
   "mappings" : {
      "document": {
            "properties" : {
               "cluster" : {
                  "properties": {
                     "id": {"type": "keyword"}
                  }
         },
               "title" : { "type" : "text"},
               "title_cleaned" : {
                           "type" : "text",
                           "index": "no"
               },
               "author" : { "type" : "text"},
               "url" : { "type" : "text",  "index": "not_analyzed"},
               "notes" : { "type" : "text"},
               "tags" : { "type" : "text"},
               "group_name" : { "type" : "keyword",  "index": "not_analyzed"},
               "group_description" : { "type" : "text",  "index":
                   "not_analyzed"},
               "group_image" : { "type" : "keyword",  "index":
                   "not_analyzed"},
               "discipline" : { "type" : "keyword"},
               "extras" : { "type" : "text"},
               "language" : { "type" : "keyword"},

               "created_at" : { "type" : "date",
               "format": "yyyyMMdd"
               }
         }
      }
   }
}
```

TABLE B.1: A JSON query to create the index in Elasticsearch

Appendix C

Test Users' Background

User	Subject	Layout	Time [min]	Task Completed*	Browser
1	Mechatronics	1	14	1	Safari
2	Mechatronics	2	14	1	Chrome
3	Bioprocess Engineering	3	22	2	Firefox
4	Computational Science	2	17	1	Chrome
5	Environmental Engineering	3	21	2	Firefox
6	Naval Architecture	1	17	2	Firefox
7	Medical Engineering	2	11	1	Firefox
8	Environmental Engineering	1	7	1	Safari
9	Mechatronics	3	17	1	Firefox
10	Technomathematics	1	17	2	Chrome
11	Computational Science	2	9	1	Chrome
12	Computational Science	1	11	1	Chrome
13	Mechatronics	3	15	2	Chrome
14	Civil Engineering	2	13	1	Firefox
15	Chemistry	1	15	2	Firefox
16	Economics	3	17	2	Safari
17	Geophysics	2	13	1	Firefox
18	Mathematics	3	18	2	Chrome
19	Chemistry	2	7	1	Chrome
20	Wood Science	1	10	1	Firefox
21	Mathematics	3	18	2	Firefox
22	Economics	2	12	1	Chrome
23	Physics	1	16	2	Firefox
24	Wood Science	3	19	2	Chrome
25	Mathematics	2	6	1	Firefox
26	Physics	1	18	2	Firefox
27	Economics	3	8	1	Safari
28	Mathematics	1	8	1	Chrome
29	Computational Science	1	14	2	Chrome
30	Mathematics	2	14	2	Chrome
31	Economics	2	15	2	Safari
32	Economics	1	14	1	Chrome
33	Computational Science	3	22	2	Chrome

*: 1 stands for task completed without help. 2 stands for task completed with help.

Appendix D

System Usability Scale

TABLE D.1: System usability scale

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

Appendix E

Scenario Tasks for Test users

- **Task 01** - Navigate to a dataset without using the keyword search field.
- **Task 02** - Navigate to datasets containing the word *radar*.
- **Task 03** - Navigate to the datasets of the group *Datacite*.
- **Task 04** - The result of task 04 would be grouped into 2 clusters. Display only datasets from one of these clusters. (only by layout 1).
- **Task 05** - Filter the result by discipline and select datasets about *Archeology*.
- **Task 06** - Select the dataset by the author *Headland Archaeology Ltd*.
- **Task 07** - Select one of the dataset and check the metadata.
- **Task 08** - Clear the filters (only by layout 1 and 2).
- **Task 09** - Navigate to datasets containing the word *radar* again.
- **Task 10** - Navigate to the datasets of the group *NARCIS*.
- **Task 11** - What is the size of the result?
- **Task 12** - Select the dataset published on 19/07/2011 (only by layout 1 and 2).
- **Task 13** - Navigate to datasets containing the words *radar* and *data*.

Appendix F

Test Users' Log

User	Layout	Time [min]	Task Completion*	Remark**
1	1	14	1	2 4 2 4 3 3 3 4 4 2
2	2	14	1	2 5 2 5 2 3 4 3 4 3
3	3	22	2	4 4 2 2 4 4 5 2 5 3
4	2	17	1	3 2 3 4 4 4 3 3 5 3
5	3	21	2	5 1 3 3 3 3 4 3 4 2
6	1	17	2	2 3 4 1 1 3 3 4 3 5
7	2	11	1	3 3 2 4 4 4 5 3 5 3
8	1	7	1	4 3 4 3 3 4 3 5 4 2
9	3	17	1	4 2 3 3 3 3 4 3 5 3
10	1	17	2	3 3 1 2 2 3 3 3 3 4
11	2	9	1	2 3 3 3 3 3 4 3 4 3
12	1	11	1	3 4 3 4 4 3 4 3 3 3
13	3	15	2	5 2 4 3 4 1 3 2 3 3
14	2	13	1	3 4 3 2 3 3 2 3 3 2
15	1	15	2	3 4 4 3 3 3 3 3 5 2
16	3	17	2	5 3 4 4 5 4 3 4 3 3
17	2	13	1	3 3 4 3 4 3 4 3 5 5
18	3	18	2	5 4 5 3 4 3 3 2 3 3
19	2	7	1	4 2 3 4 3 2 2 3 5 2
20	1	10	1	3 4 2 4 4 3 4 2 3 3
21	3	18	2	4 4 3 2 3 2 5 3 5 3
22	2	12	1	3 4 5 3 2 3 3 3 3 4
23	1	16	2	4 2 4 3 4 4 4 4 3 3
24	3	19	2	3 3 2 4 3 2 3 3 4 2
25	2	6	1	5 4 3 3 3 2 4 3 3 3
26	1	18	2	4 3 5 3 5 3 5 2 5 5
27	3	8	1	5 4 3 3 4 3 2 3 3 2
28	1	8	1	3 4 4 2 4 3 4 3 5 3
29	1	14	2	4 3 1 3 3 3 2 4 3 3
30	2	14	2	3 4 3 3 2 2 4 3 4 4
31	2	15	2	5 5 4 3 3 3 3 2 4 3
32	1	14	1	5 4 3 4 4 4 2 3 5 5
33	3	22	2	5 2 5 1 4 3 5 2 5 2

* : 1 stands for task completed without help. 2 stands for task completed with help.
**: the remark column contains the answer of the SUS questions. The numbers are ordered by question index.

Appendix G

Experiment Analysis

In [566]:
```
from statsmodels.stats.multicomp import pairwise_tukeyhsd

tukey = pairwise_tukeyhsd(endog=df['time'],     # Data
                          groups=df['layout'],   # Groups
                          alpha=0.05)          # Significance level

tukey.plot_simultaneous()    # Plot group confidence intervals
plt.vlines(x=49.57,ymin=-0.5,ymax=4.5, color="red")

tukey.summary()
```

Out[566]:

Multiple Comparison of Means - Tukey HSD,FWER=0.05

group1	group2	meandiff	lower	upper	reject
1	2	-1.0909	-4.8758	2.694	False
1	3	4.7273	0.9424	8.5122	True
2	3	5.8182	2.0333	9.6031	True

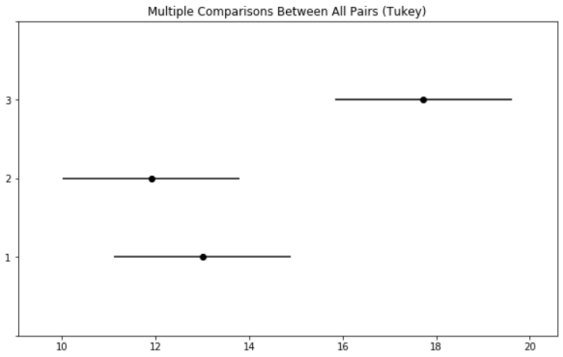


FIGURE G.1: Multiple comparison of time token by each group using Tukey's honest significance test.

C

Appendix H

CD-ROM

In the CD attached with this work are three directories:

1. **thesis/**

   This directory contains the latex source code of this research, including the figures used and some scientific papers.

2. **notebook/**

   This directory contains the Python scripts that were used to evaluate the distance metrics, implement the RS and test the performance. The scripts are in form of snippets in a notebook which can be executed under Jupyter (http://jupyter.org).

3. **source/**

   This directory contains the implementation of the layouts.

Bibliography

[1] Alessandro Moschitti; Roberto Basili. "Complex Linguistic Features for Text Classification: A Comprehensive Study" (2004).

[2] m.c. schraefel Bill Kules; Max L. Wilson and Ben Shneiderman. "From Keyword Search to Exploration: How Result Visualization Aids Discovery on the Web" (2004).

[3] Willem Bressers. "Optimizing a web search engine user interface" (2007).

[4] Gerard Salton; Christopher Buckley. "Term-weighting approaches in automatic text retrieval" (1998).

[5] Elasticsearch B.V. *Elasticsearch Reference*. elastic.co, 2018.

[6] Steven T. Kirsch; William I. Chang and Ed R. Miller. "Real-time document collection search engine with phrase indexing" (1997).

[7] Gitte Lindgaard; Jarinee Chattratichart. "Usability testing: what have we overlooked?" (2007).

[8] Sanjay Agrawal; Surajit Chaudhuri and G. Das. "DBXplorer: a system for keyword-based search over relational databases" (2002).

[9] Kilian Q. Weinberger; Fei Sha; Minmin Chen and Zhixiang (Eddie) Xu. "An alternative text representation to TF-IDF and Bag-of-Words" (2013).

[10] Jacques Bughin; Michael Chui and James Manyika. "Clouds, big data, and smart assets: Ten tech-enabled business trends to watch" (2010).

[11] Ruksana Akter; Yoojin Chung. "An Evolutionary Approach for Document Clustering" (2013).

[12] World Wide Web Consortium. *Web Content Accessibility Guidelines 1.0*. https://www.w3.org/TR/WAI-WEBCONTENT/.

[13] Divyakant Agrawal ; Sudipto Das and Amr El Abbadi. "Big Data and Cloud Computing: Current State and Future Opportunities" (2011).

[14] University of Edinburgh Data Library team EDINA. *Research Data Management*. https://blogs.ntu.edu.sg/lib-datamanagement/introduction/.

[15] MDN web docs. *An introduction to Ajax - Asynchronous JavaScript + XML*. https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX.

[16] Richard H. Fowler; Wendy A. L. Fowler and Jorge L. Williams. "3D visualization of WWW semantic content for browsing and query formulation" (1997).

[17] Johannes Fürnkranz. "A Study Using n-gram Features for Text Categorization" (2003).

[18] Mesfin Sileshi; Bjorn Gambäck. "Evaluating Clustering Algorithms: Cluster Quality and Feature Selection in Content-Based Image Clustering" (2009).

[19] Miguel Grinberg. *Flask Web Development*. O'Reilly UK Ltd.; 2nd edition edition, 2018.

[20] Mehdi Allahyari; Elizabeth D. Trippe; Juan B. Gutierrez and Krys Kochut. "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques" (2017).

[21] Chris Burges; Tal Shaked; Erin Renshaw; Matt Deeds; Nicole Hamilton and Greg Hullender. "Learning to rank using gradient descent" (2005).

[22] Khalid Majrashi; Margaret Hamilton and Alexandra L. Uitdenbogerd. "Multiple User Interfaces and Cross-Platform User Experience: Theoretical Foundations" (2015).

[23] Jason Ellis; Achille Fokoue; Oktie Hassanzadeh. "Exploring Big Data with Helix: Finding Needles in a Big Haystack" (2012).

[24] Marc Hassenzahl and Noam Tractinsky. "User experience - A research agenda. Behaviour & Information Technology" (2011).

[25] Manjunath T.N; Ravindra S. Hegadi and Ravikumar G. K. "A Survey on Multimedia Data Mining and Its Relevance Today" (2010).

[26] Craig Silverstein; Hannes Marais; Monika Henzinger and Michael Moricz. "Analysis of a very large web search engine query log" (1999).

[27] Joshua Zhexue Huang. "Clustering Categorical Data with k-Modes" (1998).

[28] Joshua Zhexue Huang. "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values" (1998).

[29] David A Hull. "Stemming algorithms: A case study for detailed evaluation" (1996).

[30] The European Data Infrastructur. *B2DROP*. https://eudat.eu/services/b2drop.

[31] The European Data Infrastructur. *B2FIND*. https://eudat.eu/services/b2find.

[32] The European Data Infrastructur. *B2SAFE*. https://eudat.eu/services/b2safe.

[33] The European Data Infrastructur. *B2SHARE*. https://eudat.eu/services/b2share.

[34] The European Data Infrastructur. *B2STAGE*. https://eudat.eu/services/b2stage.

[35] The European Data Infrastructur. *Deutsche Klimarechenzentrum*. https://www.dkrz.de/up/services/data-management/projects-and-cooperations/eudat-b2find-catalogue.

[36] The European Data Infrastructur. *EUDAT Services for Data Preservation*. https://eudat.eu/data-preservation.

[37] Thorsten Joachims. "Text categorization with support vector machines: learning with many relevant features." (1998).

[38] Thorsten Joachims. *Text categorization with Support Vector Machines: Learning with many relevant features*. LNCS, 2005.

[39] Jane Radatz; Anne Geraci; Feny Katki and John Lane. "IEEE Standard Glossary of Software Engineering Terminology" (2007).

[40] Heide Brücher; Gerhard Knolmayer and Marc-André Mittermayer. "Document Classification Methods for Organizing Explicit Knowledge" (2002).

[41] Marek Rogozinski; Rafal Kuc. *Elasticsearch Server*. PACKT PUBLISHING.

[42] James Tin yau Kwok. "Automated Text Categorization Using Support Vector Machine" (1998).

[43] Aurangzeb Khan; Baharum Baharudin; Lam Hong Lee and Khairullah khan. "A Review of Machine Learning Algorithms for Text-Documents Classification" (2010).

[44] Daniel E. Rose ; Danny Levinson. "Understanding User Goals in Web Search" (2015).

[45] David D. Lewis. "Evaluating Text Categorization" (1998).

[46] Chih-Wei Hsu; Chih-Jen Lin. "A comparison of methods for multiclass support vector machines" (2002).

[47] Tie-Yan Liu. "Learning to Rank for Information Retrieval" (2009).

[48] Yiming Yang; Xin Liu. "A re-examination of text categorization methods" (1999).

[49] Gary Marchionini. "Information Seeking in Electronic Environment" (1996).

[50] Sam Scott; Stan Matwin. "Feature Engineering for Text Classification" (2011).

[51] Ji-Rong Wen; Jian-Yun Nie and Hong-Jiang Zhang. "Clustering user queries of a search engine" (2001).

[52] Andrew McCallum; Kamal Nigam. "A Comparison of Event Models for Naive Bayes Text Classification" (1998).

[53] Stuart Russell; Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[54] MGreg Nudelman. "Designing Search: UX Strategies for eCommerce Success" (2005).

[55] Sergey Brin ; Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine" (1998).

[56] Jasmina Dj. Novakovic; Alempije Veljovic; Sinisa S. Ilic; Zeljko Papi and Milica Tomovic. "Evaluation of Classification Models in Machine Learning" (2017).

[57] Rosenfeld Peter Morville; Louis. "Information Architecture for the World Wide Web: Designing Large-Scale Web Sites" (2016).

[58] David Powers. "Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation" (2011).

[59] N. Sunil Chowdary; D. Sri Lakshmi Prasanna and P. Sudhakar. "Evaluating and Analyzing Clusters in Data Mining using Different Algorithms" (2014).

[60] Christopher D. Manning; Prabhakar Raghavan and Hinrich Schütze. "Introduction to Information Retrieval" ().

[61] Wolfgang E. Nagel Richard Grunzke Volker Hartmann. "Towards a Metadata-driven Multi-community Research Data Management Service" (2016).

[62] I. Rish. "An empirical study of the naive Bayes classifier" (2002).

[63] Greg Smith; Mary Czerwinski; Brian Meyers; Daniel Robbins; George Robertson and Desney S. Tan. "FacetMap: A Scalable Search and Browse Visualization" (2006).

[64] Stephen Robertson. "Understanding Inverse Document Frequency: On theoretical arguments for IDF" (2004).

[65] Magnus Rosellk. “Introduction to Text Clustering” (2008).

[66] Ashutosh Garg; Dan Roth. “Understanding Probabilistic Classifiers” (2001).

[67] Peter J. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis” (1986).

[68] Miriam; He Yulan Saif Hassan; Fernández and Harith Alani. “On stopwords, filtering and data sparsity for sentiment analysis of Twitter” (2014).

[69] Mark van de Sanden; Christine Staiger; Roberto Mucci; Stephane Coutin; Hannes Thiemann. “Final Report on EUDAT Services”. *EGU General Assembly 2016*, (2015).

[70] Kiri Wagstaf; Claire Cardie; Seth Rogers; Stefan Schroedl. “Constrained K-means Clustering with Background Knowledge” (2001).

[71] Fabrizio Sebastiani. “Machine Learning in Automated Text Categorization” (2008).

[72] Mark H. HansenElizabeth A. Shriver. “Method for organizing records of database search activity by topical relevance” (2001).

[73] Viknes Balasubramanee; Chathuri Wimalasena; Raminder Singh and Marlon Pierce. “Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development” (2013).

[74] Taufik Akbar Sitompul. *Usability and User Experience Evalu- ation of EUDAT Services*. 2016.

[75] International Organization for Standardization. “Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs), Part 11: Guidance on Usability” (1998).

[76] International Organization for Standardization. “Human-centred Design for Interactive Systems: ISO” (2010).

[77] P.N. Tan; Michael Steinbach and Vipin Kumar. “Cluster Analysis: Basic Concepts and Algorithms” (2005).

[78] D. T. Pham; Maria Mar Suarez-Alvarez and Yuriy I. Prostov. “Random search with k-prototypes algorithm for clustering mixed datasets” (2011).

[79] Mehdi Allahyari; Seyedamin Pouriyeh; Mehdi Assef; Elizabeth D. Trippe and Juan B. Gutierrez. “A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques” (2017).

[80] San Diego Vagelis Hristidis; UC. “Efficient IR-style keyword search over relational databases” (2003).

[81] Vapnik Vladimir. *The Nature of Statistical Learning Theory*. Springer, 1995.

[82] Jonathan J. Webster and Chunyu Kit. “Tokenization as the initial phase in NLP” (1992).

[83] Hannes Widmann Heinrich; Thiemann. “EUDAT B2FIND : A Cross-Discipline Metadata Service and Discovery Portal”. *EGU General Assembly 2016*, (2016).

[84] the free encyclopedia Wikipedia. *Ajax - Asynchronous JavaScript + XML*. https://en.wikipedia.org/wiki/Ajax_(programming).

[85] Wikipedia, the free encyclopedia. *A tag cloud with terms related to Web 2.0*. Original by Markus Angermeier Vectorised and linked version by Luca Cremonini. 2007. URL: https://en.wikipedia.org/wiki/Tag_cloud#/media/File:Web_2.0_Map.svg.

[86] Yiming Yang. "An Evaluation of Statistical Approaches to Text Categorization" (1999).

[87] Yuh-Jye Lee; Yi-Ren Yeh; and Hsing-Kuo Pao. "An Introduction to Support Vector Machines" (2005).

[88] Charu C. Aggarwal; ChengXiang Zhai. "A Survey of Text Clustering Algorithms" (2012).

[89] Michal Jankowski-Lorek; Kazimierz Zielinski. "Document controversy classification based on the wikipedia category structure" (2015).