MASTER

Visualization for text-based research in philosophy

Hofstede, S.E.

Award date:
2018

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
Visualization for Text-based Research in Philosophy

Master Thesis

Steven Hofstede

Supervisors:
Michel Westenberg
Thom Castermans
Rob Koopman

Final version

Eindhoven, October 2018
Abstract

Conceptual analysis and close reading form an important part of philosophy research. A researcher focuses on an individual text passage from a collection of works to analyse its full interpretation. This can be a very time-consuming process if this is done manually since it is often also involved with complex work that have a high conceptual density. The analysis can lead to the generation of hypotheses based on contradicting text passage interpretations.

This thesis introduces BolVis, an interactive visualization tool for text-based research in philosophy that can be used to accept or reject these hypotheses. We designed and implemented a prototype that uses semantic similarity search on individual words, phrases and sentences. The visualization allows investigators to quickly discover the most relevant parts of the corpus corresponding to their search query and provides functionalities that makes the exploration of the semantic context easier. We apply our tool to a corpus containing the works of polymath Bernard Bolzano (1781-1848) consisting of around 11,000 pages to show the benefits of BolVis.
Acknowledgements

I would like to thank my graduation supervisor Michel Westenberg for his guidance and his aid during this project. His feedback, advice and comments on both the report and the project have helped me improve my scientific research skills.

I would also like to thank my technical advisor Thom Castermans for his help with the technical parts of this project and for all his advices in writing this thesis. His skills with Java, HTML, CSS, and Javascript have proven to be essential in solving most technical problems.

Furthermore, I would like to thank Shenghui Wang and Rob Koopman, my supervisors at OCLC. They have provided me different insights in problem solving and their comments and feedback have helped me improve my ideas.

In addition, I would also like to thank Arianna Betti, Pauline van Wierst and Yvette Oortwijn, researchers at the UvA, for their views on the philosophy side of this project. Also their feedback has helped me refine parts of the application.

Last but not least, I would like to thank my family for giving me the opportunity to graduate from the Technical University of Eindhoven and for their endless support during these college years.
# Contents

<table>
<thead>
<tr>
<th>Contents</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 <strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem description</td>
<td>1</td>
</tr>
<tr>
<td>1.2 The Bolzano dataset</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Additional data</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Organization</td>
<td>4</td>
</tr>
<tr>
<td>2 <strong>Related work</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.1 Set intersections</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Close and distant reading</td>
<td>6</td>
</tr>
<tr>
<td>3 <strong>User Requirements</strong></td>
<td>10</td>
</tr>
<tr>
<td>3.1 General requirements</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Capability requirements</td>
<td>10</td>
</tr>
<tr>
<td>3.3 Constraint requirements</td>
<td>11</td>
</tr>
<tr>
<td>3.4 Use cases</td>
<td>12</td>
</tr>
<tr>
<td>4 <strong>Design decisions</strong></td>
<td>13</td>
</tr>
<tr>
<td>4.1 Passage windows</td>
<td>13</td>
</tr>
<tr>
<td>4.2 Final prototype design</td>
<td>14</td>
</tr>
<tr>
<td>4.3 Revised decisions</td>
<td>16</td>
</tr>
<tr>
<td>5 <strong>Prototype implementation</strong></td>
<td>19</td>
</tr>
<tr>
<td>5.1 Ariadne</td>
<td>19</td>
</tr>
<tr>
<td>5.1.1 Combining off-line and online processes</td>
<td>19</td>
</tr>
<tr>
<td>5.1.2 Current interface</td>
<td>20</td>
</tr>
<tr>
<td>5.2 Preprocessing data</td>
<td>22</td>
</tr>
<tr>
<td>5.3 Implementation</td>
<td>25</td>
</tr>
<tr>
<td>5.4 BolVis</td>
<td>26</td>
</tr>
<tr>
<td>5.4.1 Book Structure View</td>
<td>26</td>
</tr>
<tr>
<td>5.4.2 Text Passage View</td>
<td>27</td>
</tr>
<tr>
<td>5.4.3 Favorites View</td>
<td>29</td>
</tr>
<tr>
<td>6 <strong>Conclusions</strong></td>
<td>31</td>
</tr>
<tr>
<td>6.1 Evaluation</td>
<td>31</td>
</tr>
<tr>
<td>6.2 Conclusion</td>
<td>33</td>
</tr>
<tr>
<td>6.3 Future work</td>
<td>34</td>
</tr>
<tr>
<td><strong>Bibliography</strong></td>
<td>35</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Problem description

Researchers at the philosophy department of the Universiteit van Amsterdam (UvA) spend much of their research time on performing conceptual analysis and close reading of philosophical texts. The latter is a very accurate process where an investigator focuses on a single specific text passage or philosophical notion from one or multiple philosophical works (oeuvres) of an author. Traditional close reading is, however, very time-consuming and forms high restrictions on the scalability of this conceptual analysis as the written work is very complex with a high conceptual density. This results in hypotheses for the analysis about contradictory concepts of these text passages, that are either accepted or rejected based on the interpretation of the text [9].

One of the authors of these studied texts is Bernard Bolzano (1781-1848), a mathematician, logician, philosopher, and theologian who wrote several mathematical and philosophical articles in German. For the complete works of Bolzano, the UvA currently uses a tool with a basic user interface that is developed by the Online Computer Library Center (OCLC). This tool supports semantic search over the whole collection of the works of Bolzano, based on word embeddings and document embeddings. The goal of this project is to design and implement an interactive web-based tool which allows the users to explore the complete works of Bolzano where they can use the visualization to reliably identify sentences, paragraphs and chapters that contain a combination of one or more concepts.

This document contains the analytical process and technical details in developing an application we call BolVis. This webapplication meets the requirements that have been established by the author based on meetings and discussions with Rob Koopman and Shenghui Wang from the research department of the OCLC research office and Arianna Betti, Pauline van Wierst and Yvette Oortman from the philosophy department of UvA. The visualization of this application is made as part of the 2IMC00 Graduation Project at Eindhoven University of Technology.

1.2 The Bolzano dataset

The Bolzano dataset, established in 2017, contains mathematical and philosophical texts written by Bernard Bolzano (1781-1848) in German and has been made available by Arianna Betti and Yvette Oortwijn (UvA). It consists of books as well as manuscripts and letters. This dataset consists of folders with OCR-ed scans of printed material in DjVu, PDF and text format. Optical Character Recognition (OCR) is the transformation of an image that contains text into editable electronic text by recognizing and storing the patterns of all characters. All folders, with the exception of two, are named according to a corresponding book ID and contain three files, (DjVu, PDF and text), where each file corresponds either to a whole book or to one section of a book. All DjVu files in the dataset are also present in text format and PDF format, but not vice versa. All these books were published as part of Bolzano’s collection, named the *Bernard Bolzano Gesamtausgabe*. 
CHAPTER 1. INTRODUCTION

(BGA). The scanning and revising of the BGA has been done by an expert in OCR with a precision of 99%. The dataset is not processed for text analyses, meaning that the files reproduce as closely as possible to the printed volumes. This means that the volumes containing letters, for instance, are not split by letter, while a letter counts in fact as one single document. The so called added parts such as the prefaces and notes by other interpreters who are not Bolzano himself are also contained in the files. For some analyses this can be problematic. This is a pressing problem since the UvA and OCLC generally do not have the resources to manually process the datasets.

The two folders that differ from the rest are regarding the *Paradoxien des Unendlichen* and the *Wissenschaftslehre* books. The original *Paradoxien des Unendlichen* has not been published as part of the BGA. The available edition has been scanned and OCR-ed by Ocular and by the UvA and OCLC, with suboptimal results. This book is split into sections, and what we would consider ‘added’ parts to the works such as the introduction by the editor and the index. These added parts have been removed. The *Wissenschaftslehre* files are obtained by scanning and OCR-ing the BGA edition by a trainee, the text has been split into sections and used for text analysis applying information retrieval techniques before. In this document we refer to the Bolzano dataset as all books that were published as part of the BGA, plus the two books *Wissenschaftslehre* and *Paradoxien des Unendlichen*.

Some parts of the texts in the corpus contain (mathematical) formulas and/or symbols that have been replaced by Latin characters in the OCR-process. This resulted in character strings that are left unchanged and could have a small influence to the outcome of computational text analysis. End-of-line hyphenation and emphasis of words are not removed either and could also influence the outcome of computational text analysis.

The aim of the creation of the Bolzano dataset is to trace the development of certain notions such as *analytic sentences* or in German *analytische Sätze*. Furthermore, for the quantitative testing of certain hypotheses, such as whether analyticity is related to the notion of *Abfolge*.

1.3 Additional data

Additional data has been provided by OCLC that corresponds with the Bolzano dataset. The OCR scans of the written texts have been divided into sentences where the correspondence of each sentence has been annotated by hand among the works, the hierarchical structure of the works, and the pages. The goal of this additional data is to provide extra information on the textual context so that the Bolzano dataset can be used for a broad analysis. It consists of the following files.

- **vocab**: contains a list of most frequent terms or words occurring in the Bolzano dataset. Each word that occurs somewhere in the corpus is listed in this Tab Separated Values (TSV) file and is sorted on the number of occurrences shown in the second column of each of these words. This is followed by the weight of that word where the higher its importance, the closer it is to 1, and vice versa closer to 0. The weight and the number of occurrences are correlated but not absolute.

- **vector**: binary file that can be considered as a matrix that holds vector information of the words in the corpus and is used in the calculation of the relevance between words. This file originally starts with the number of lines in the file **vocab** with the size of the vectors.

- **docvocab**: file where each line holds the information for a sentence in the corpus. This information is the offset of that sentence in the file followed by its length, ID, and year of that sentence respectively. Each line in this file corresponds with the sentence on the same line number in the file **bolzano_sentences_20180608.txt** as you can see in Figure 1.1.

- **docvector**: file that is similar to **vector** but instead it is concerned with sentences instead of words.
CHAPTER 1. INTRODUCTION

- bolzano_sentences_20180608.txt: TSV that contains all sentences in the corpus where each sentence has been given an ID in the following structure:

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>9783772804618_1_page22_sect168_sent12</td>
<td>[title:wir wollen nur zeigen da auch die heil]</td>
</tr>
<tr>
<td>9783772804557_0_page23_sectVORREDE_sent46</td>
<td>[title:wir haben sie ganz weggelassen]</td>
</tr>
</tbody>
</table>

In the first part before the tab, the ID of the book where that sentence is stated is given before the underscore. Then we see the value 1. This value can either be 1 or 0. For type 0 it means that the sentence is separated by sections. For type 1, they are also individual parts of the same book but not segmented by numerical sections. They are named after the first word (e.g. VORREDE in this case) or 0, where the page number is the beginning page of that part. This type value is then followed by the page number, section name/number, and sentence number. After the tab, the sentence itself is shown but stripped from their punctuation marks. Each line in this file corresponds to the sentence information on the same line number in docvocab as you can see in Figure 1.1.

There are a few books in this file where the type value and the page number is missing from the ID. These books were segmented in a different way by OCLC since these books are structured differently than the other books in the corpus. These books are Wissenschaftslehre, Paradoxien des Unendlichen, and Philosophische Tagebücher and have the following structure in the data:

```
nameOfBook_sectX_paraY_sentZ → [title:S]
```

So this structure starts with the name of the book, i.e., either Wissenschaftslehre, Paradoxien des Unendlichen, or band17_1817-1827 respectively. This is followed by a section X, a paragraph Y, and a sentence Z, where X,Y,Z ∈ N, with the corresponding sentence S.

- sentences_original_20171118.txt: TSV file that is very similar to bolzano_sentences_20180608.txt where it holds the same ID’s with the same sentences except that these sentences are original in the sense that they are not stripped from their punctuation marks and are derived directly from the book.

- tab.txt: TSV file that holds the hierarchical structure of the books. This structure looks like the following, where the arrows indicate a tab:

```
9783772804557/23 → Book title : RW1 1-85 → Chapter : Lehrbuch der Religionswissenschaft → Subchapter : Vorrede → Year : 1840-1845
```

First we see the ID of this book structure that consists of the ID of the book followed by the starting page number behind the “/”. In this case we can see that the subchapter Vorrede starts at page 23 of the book RW1 1-85 and belongs to chapter Lehrbuch der Religionswissenschaft. This page number is consistent with the page numbers in the ID’s of bolzano_sentences_20180608.txt. So for our example stated above, the sentence [title:wir haben sie ganz weggeallsen] is stated in this particular subchapter of the book RW1 1-85. Finally, we can see that this specific line has a Year attribute, containing either a year range or a single year of which that book is written. This attribute is not present for every book in the data.
CHAPTER 1. INTRODUCTION

1.4 Organization

This thesis is structured in the following way. First, we will discuss related work to this project in Chapter 2. Second, the established user requirements are shown in Chapter 3. Then we will talk about several design decisions we made in Chapter 4. This is followed by a chapter that provides a detailed explanation about the implementation of the prototype we call BolVis. This can roughly be interpreted as the visualization for the Bolzano tool and is discussed in Chapter 5. After the research and reasoning behind this visualization, we will finally make some conclusions about our work and evaluate the project in Chapter 6.
Chapter 2

Related work

2.1 Set intersections

In visualizing the exploration of the philosophical corpus we are most interested in certain sets of results and particular in the relation between them. Understanding this relationship is an important part of analysis tasks. The most common visualization method for showing these relationships are Euler and Venn diagrams where Venn diagrams are a type of Euler diagram. Every set is often represented as a circle that show all possible logical relations between these finite number of sets where Venn diagrams also depict the empty intersections whereas Euler diagrams do not. Figure 2.1 shows two examples of Venn diagrams, one that displays the intersections between three sets in Figure 2.1a and one that depicts them for five sets in Figure 2.1b.

![Venn diagram with 3 sets](image1.png)

![Venn diagram with 5 sets](image2.png)

Figure 2.1: Venn diagrams become more difficult to analyse with higher number of sets.

The size of the surface is then often represented as the size of the sets and their logical relations, an area proportional visualization technique, or it just illustrates and labels the sets, as in the above figure. From it we can see that the relations between a small number of sets is easy but becomes exceedingly more challenging as the number of sets increases. Examples of recent works that use these types of diagrams are in [1] and [6] where the authors use five- and six-set Euler diagrams to show the intersections of the genomes of a pine in the first and a banana in the second work. In both works, it becomes more difficult to see which sets are joint in which intersection for sets that large.

UpSet [2], a visualization tool that is used for the quantitative analysis of large sets, addresses this issue. The technique uses a matrix layout to visualize set intersections that provides an efficient way to view related data, like the number of elements it contains (the cardinality), the
deviation, and summary data of element attributes. In Figure 2.2 we can see an example of UpSet on a movie genres data set that contains 17 sets with 4 attributes. All possible intersections are shown in the Combination Matrix where each row represents an intersection that is equivalent to an area in a Venn diagram and each column represents the set. The cells in the matrix are either black, meaning that the set takes part in the intersection, or light-gray, explaining that it does not. The first row in Figure 2.2 shows only a black dot for the comedy set while the rest are light-gray, indicating that this (single) set has 1003 elements. The (selected) second row visualizes the intersection of the sets action and comedy, meaning that there are 38 movies that both have these move genres. In the additional columns from the figure we can deduce that the selected set has been watched more often than the adventure movies. It is possible to sort on the related data, including on each of the associated attributes so that it allows for an effective analysis of intersections and aggregates that are relevant.

2.2 Close and distant reading

As we mentioned in Section 1.1, conceptual analysis and close reading are fundamental parts of research in philosophy. With close reading, a researcher is analyzing a single text passage to discover its full interpretation [22]. This analysis includes close investigation of used words and phrases, style and syntax of the text, but also considering the authors, ideas, and occurrences. In Figure 2.3a we can see an example of how traditional close reading is done by hand. Different kinds of methods are used to manually add notes to the text and making links between various parts of the passage. Some of these methods are underlining words, phrases and/or sentences that seem important in several styles and using different colors and is thus a very time-consuming process.

On the other hand, there is distant reading that does the precise opposite. With distant reading, a researcher tries to create a more abstract view of a text passage by looking at the textual content and change it into more global characteristics. Nowadays, books and texts are largely digitally available and this opens up new opportunities for close and distant reading. There are various visualization tools for conceptual analysis that supports both close and distant reading where Jänicke et al. [22] provide an overview that shows the research papers of these tools.
CHAPTER 2. RELATED WORK

(a) Example of how close reading is done manually in practice [11].


Figure 2.3

using a classification. Single text analysis, Parallel text analysis and Corpus analysis are the three categories used in their classification. The first focuses on the analysis and visualization of a single literary work while the second category aims at displaying similarities and differences between equivalent text passages or literary work. With corpus analysis, the visualization is focused on a large number of works from an individual corpus. The authors list all visualization techniques for close and distant reading for each category. Since our approach focuses on formulating hypotheses about conflicting interpretations of a single text passage within the entire corpus of a single author, we need to look at the combination of single text analysis and corpus analysis.

The most common technique used for this type of analysis combination is visualizing features of textual entities of passages by using coloring and different font sizes to display several types of information of a word or phrase. Tools using this technique often implement a word cloud to show summary statistics of the text and support close reading using highlighting or connections to link related annotations. Viegas, Wattenberg, and Feinberg, 2009 [7] adopted the word cloud design to show the significance of a word in relation to its corresponding corpus. In Figure 2.3b we see how the tool of Viegas et al., 2009, is used to portray the campaign candidates of the American elections in 2008 with a word cloud that relates to the Republican and Democratic campaign blogs.

For a large number of documents in the corpus, many other works use graphs to give an overview of the corpus and to visualize the relationships between texts based on content similarity. One of these works is Overview, a tool presented by Brehmer, Ingram, Stray, and Munzner, 2014 [14] that uses document mining on a large collection of investigative journals. Each journal is converted to a words vector with a corresponding weight that is determined by the Term Frequency-Inverse Document Frequency (TF-IDF). The similarity between the journals is calculated by the cosine distance and is used to create a hierarchical clustering that is ultimately visualized as a tree. The TF-IDF scores are then used to extract keywords that are labeled to each cluster. In Figure 2.4 we can see how Overview visualizes a tree based on a document collection of White House email messages regarding the 2010 BP oil spill. A user is able to navigate through this tree, find clusters, and read single documents where they can be annotated with user-defined tags that can be assigned individually or at the cluster level. Looking at Figure 2.4 again, we see that the “Obama letter” tag is selected in green and the corresponding clusters are highlighted with an identical tag in the same color. The keywords of the selected cluster in the third row of the graph are visualized in a list on the right. Then selected an entry on this list will display the original
document of where that keyword is stated. The downside of this tool is that it only allows the search for individual keywords while we also need direct search of the entire text.

Figure 2.4: Overview, a web application that visualises the hierarchical clustering of a document collection as a tree. [14].

There are only a few other works that combine close and distant reading that are also suited to our needs. Metatation [8] is a tool that uses annotations to guide distant reading of a literary corpus during the close reading of an individual text passage. The annotations that users make when performing a close reading are used as implicit cues for finding and creating additional metadata that is relevant for their current workflow in real-time. The tool then uses this information to generate query results that are relevant in the context of the interesting corpus. The goal of this tool is to generate meanings of literary text based on the analysis of how linguistic features co-operate while our approach focuses more on active reading of scientific material to comprehend the text arguments. This means that their method is not easily adaptable to be used for our purposes.

Another tool in the area of corpus analysis is Sketch Engine [12], a system that identifies word behaviour in various contexts. It offers a large variety of analytic functions on large collections of corpora. Kocincová et al. 2015, describe in their paper the two main functions of Sketch Engine; word sketches and a distributional thesaurus. A word sketch shows the collocational behaviour that is computed using a high number of corpus queries. A thesaurus shows a list of words that share similar collocates in corresponding grammar relations, together with their frequency and similarity score as shown in Figure 2.6a. Both main functions can be visualized as a table as displayed in Figures 2.5 and 2.6. Each column in Figure 2.5 represents one grammar relation containing collocates for the lemma strategy with the highest scores on top. Sketch Engine can also visualize these tables in circles as in Figure 2.6b where this visualization represents the same data as in Figure 2.6a. The core of this circle contains the lemma strategy and the corresponding words are placed around it based on the score; the higher the score, the closer it is placed to the lemma. The color of the circles provides a way to indicate which words are close to each other, meaning that they have similar similarity scores. The size of the circles and the font visualizes the height of the frequency of a word. However, Sketch Engine performs the search only on each separate keyword, as in the Overview tool.
### Figure 2.5: Word sketches is one of the main functions of Sketch Engine. This table shows the word sketches for lemma **strategy**.

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>approach</td>
<td>3,769,792</td>
</tr>
<tr>
<td></td>
<td>policy</td>
<td>8,292,528</td>
</tr>
<tr>
<td></td>
<td>plan</td>
<td>5,423,532</td>
</tr>
<tr>
<td>6</td>
<td>practice</td>
<td>5,116,641</td>
</tr>
<tr>
<td>5</td>
<td>management</td>
<td>7,185,783</td>
</tr>
<tr>
<td>7</td>
<td>development</td>
<td>11,306,101</td>
</tr>
<tr>
<td>8</td>
<td>initiative</td>
<td>2,562,793</td>
</tr>
<tr>
<td>9</td>
<td>solution</td>
<td>4,034,623</td>
</tr>
<tr>
<td>10</td>
<td>modal</td>
<td>5,383,962</td>
</tr>
<tr>
<td></td>
<td>method</td>
<td>4,023,295</td>
</tr>
</tbody>
</table>

(a) Distributional thesauruses are also visualized as tables where this one shows this distributional thesaurus for the same lemma **strategy**.

(b) Visualization of a thesaurus with circles for the same lemma that is now depicted in the core of the circles.

### Figure 2.6: A distributional thesaurus is the other main function of Sketch Engine.
Chapter 3

User Requirements

All capability and constraint requirements of the visualization of the Bolzano tool are stated in this chapter. This visualized tool will now be called BolVis. The visualization will satisfy these requirements. In this chapter we will first discuss the general requirements in Section 3.1. Then we will talk about the capability requirements, the requirements that fulfills an organizations role, function, and/or mission in current or future operations and are stated in Section 3.2. There are also requirements that have a restriction on the degree of freedom we have in providing a solution. These are called the constraint requirements and are shown in Section 3.3. We conclude this chapter with a list of actions or event steps typically defining the interactions between a user and a system to achieve a certain goal in Section 3.4. All requirements and use cases are consulted with the client and are given priorities according to the MoSCoW method [5]. This model is explained as follows where the capital letters in MoSCoW stand for:

M  Must have; These requirements are essential for the product.
S  Should have; These requirements are not critical for the product to work, but are nearly as important as the Must have requirements, meaning that they will be implemented if possible.
C  Could have; These requirements are desirable but not necessary to be implemented in the application. They will be included if time permits this.
W  Won’t have; These requirements will not be implemented in the application, but it would be nice if they would be implemented in the future.

3.1 General requirements

It is expected that BolVis gets a nice user interface that looks good and is also easy to understand. It’s design and interface should correctly represent the information without distortions but shows things that are clear. It should furthermore be interactive and provide a visual overview and be insightful for its users. The background of the users of the tool can typically be described as philosophers. It is expected that it will be non-collaborative, but a small number of multiple concurrent users will use it at the same time via internet.

3.2 Capability requirements

UR1  Must have
After performing a certain search query, the user must be able to see in which book, chapter, section, paragraph and sentence each of the results can be found in the entire corpus of Bolzano.
## CHAPTER 3. USER REQUIREMENTS

<table>
<thead>
<tr>
<th>User Requirement</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UR2</td>
<td>Must have</td>
</tr>
<tr>
<td>The user must be able to choose the order in which the books are shown, for instance by logical and chronological order instead of by weight/relevance.</td>
<td></td>
</tr>
<tr>
<td>UR3</td>
<td>Must have</td>
</tr>
<tr>
<td>The user must be able to choose which parts of the corpus she wants to include or exclude in the result of the query either by selecting one or more works or parts of them or by selecting a type of text or by selecting a time range.</td>
<td></td>
</tr>
<tr>
<td>UR4</td>
<td>Must have</td>
</tr>
<tr>
<td>After performing a certain search query, the user must be able to perform a search within the context of this query.</td>
<td></td>
</tr>
<tr>
<td>UR5</td>
<td>Must have</td>
</tr>
<tr>
<td>After performing multiple search queries, the user must be able to see what the results were in each of the previous search queries.</td>
<td></td>
</tr>
<tr>
<td>UR6</td>
<td>Must have</td>
</tr>
<tr>
<td>The user must be able to select or deselect the passages she wants or does not want to display for close reading and comparison.</td>
<td></td>
</tr>
<tr>
<td>UR7</td>
<td>Must have</td>
</tr>
<tr>
<td>After performing multiple search queries, the user must be able to see whether there are similarities or overlap in the search results on the chapter level, section level or paragraph level.</td>
<td></td>
</tr>
<tr>
<td>UR8</td>
<td>Should have</td>
</tr>
<tr>
<td>After performing a certain search query, the user must be able to view the original DjVu, PDF or text file for each of the corresponding passages.</td>
<td></td>
</tr>
<tr>
<td>UR9</td>
<td>Could have</td>
</tr>
<tr>
<td>After performing a certain search query, the user must be able to see a few sentences before and after each search result.</td>
<td></td>
</tr>
<tr>
<td>UR10</td>
<td>Could have</td>
</tr>
<tr>
<td>In viewing the original DjVu or PDF file for each corresponding passage, display the exact place in the book of where the passage is stated.</td>
<td></td>
</tr>
<tr>
<td>UR11</td>
<td>Won’t have</td>
</tr>
<tr>
<td>In viewing the original DjVu or PDF file for each corresponding passage, highlight the exact place in the book of where the passage is stated.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.3 Constraint requirements

<table>
<thead>
<tr>
<th>User Requirement</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UR12</td>
<td>Must have</td>
</tr>
<tr>
<td>The application must return the results within 0.5 seconds in 95% of the time.</td>
<td></td>
</tr>
<tr>
<td>UR13</td>
<td>Must have</td>
</tr>
<tr>
<td>The application must support at least 5 concurrent users that use the application at the same time.</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 3. USER REQUIREMENTS

**UR14**
BolVis must at least have the same functionalities as Ariadne.  

**UR15**
The Bolzano tool is supported on devices other than desktop computers or laptops.  

**UR16**
The language of the corpus is in German and can be changed into another language.  

**UR17**
The application searches in the corpus of other authors than Bernard Bolzano.  

**UR18**
The application is supported on other web browsers than Google Chrome.

### 3.4 Use cases

**UC1**

<table>
<thead>
<tr>
<th>Goals</th>
<th>Visual overview of the book structures and search within context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Must have</td>
</tr>
<tr>
<td>Steps</td>
<td>The user first (a) enters <em>allgemeine</em> as search query in the search bar and press search; then (b) select the option of searching within the results [restrict the search to a context] and enter <em>Grund</em> as search query in the search bar and press search; (c) restrict the results shown to paragraphs 700-708 of the Wissenschaftslehre, (d) order them logically (in ascending ordered from <em>n</em> to <em>n</em> + <em>x</em>). BolVis returns a window with a visual overview of all the relevant books/paragraphs for the query, a window with a quick passage overview and is able to get a window that displays the context of the top passage. Arianna and Pauline can now extract information from the first window as to where in the corpus they can find the result(s) of the query under consideration. They are able to switch between the results of the first query, the result of the second query and then both together. Arianna and Pauline now repeat all the steps (a)-(b) but now they add another nested query: (b-1) <em>einfach</em> before passing to restricting the results at (c). He can switch to both of the other windows to switch from what you are searching in to what you are searching exactly. Finally, the user performs another search within the context of one of the search results and gets a new list of results.</td>
</tr>
</tbody>
</table>

**UC2**

<table>
<thead>
<tr>
<th>Goals</th>
<th>Seeing previous search queries and their overlap with fast performance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Must have</td>
</tr>
<tr>
<td>Steps</td>
<td>The user enters multiple search queries in the search bar, pressing search each time. The Bolzano application returns and visualizes a list of results for each search query, each time within a couple of seconds. The user can switch between the results of each search query. The user can press a button to compare them. The application visualizes the overlap between two concepts by visualizing important calculations and intersections. The user can now perform quantitative analysis to extract the information on whether the queries have overlap.</td>
</tr>
</tbody>
</table>
Chapter 4

Design decisions

First we came up with several ideas for a prototype, given the user requirements discussed in Chapter 3. This chapter describes some of these ideas and explains the decisions we made for our final prototype design that we chose to implement. All prototypes were designed with Moqups[15], a web application that allows users to make and collaborate on wireframes, mockups, diagrams and prototypes.

4.1 Passage windows

The first prototype we created is depicted in Figure 4.1. The idea was that if the user entered a search query, for instance abfolge as we see in Figure 4.1, then he would see a words list on the left with passage windows on the rest of the screen. This words list has the same purposes and functionalities as the words list of Figure 5.3a described in Section 5.1. In addition, the horizontal colored bars would give a visual representation of the relevance value compared to the query (in green) and the number of occurrences this word has in the Bolzano dataset (in orange). The passage windows visualizes the results list belonging to that query, giving a window with

Figure 4.1: The first prototype where each result is visualized in passage windows.

The first prototype we created is depicted in Figure 4.1. The idea was that if the user entered a search query, for instance abfolge as we see in Figure 4.1, then he would see a words list on the left with passage windows on the rest of the screen. This words list has the same purposes and functionalities as the words list of Figure 5.3a described in Section 5.1. In addition, the horizontal colored bars would give a visual representation of the relevance value compared to the query (in green) and the number of occurrences this word has in the Bolzano dataset (in orange). The passage windows visualizes the results list belonging to that query, giving a window with
information for each passage. Each window begins with a detailed description of the whereabouts of that text passage, like book title, section, paragraph and so on, with the corresponding summary statistics shown as vertical colored bars. The text passage itself is then shown by highlighting that corresponding sentence in a PDF-like environment since most books are available in PDF. The user would be able to scroll through this context and perform another search query within this context. This new query-within-context search would then lead to the creation of a new passage window containing corresponding results. The association of these passage windows would be indicated with a blue arrow to the right, as seen in Figure 4.1.

This design may cover the capability requirements UR1, UR2, UR4, UR8, UR9, UR10, and UR11 mostly, but it misses some critical parts and it does not meet the other requirements. In case there are multiple results from several books, it is desirable to get an overview of the hierarchical structure of all books that visualizes where the results are from. We chose to hold the hierarchical book structure in an expansion panel because displaying it entirely would be very large. Such component reduces the amount of information that is shown to provide a better overview for the user and is useful for displaying only the content that the user needs to see.

For multiple queries, we are interested in how the results were in the previous query (UR5) and how these queries are related to each other. Therefore, we decided to use the tabs component from Angular to represent each query where every tab will have its own associated view content. This makes it very easy to switch between the contents of multiple queries.

The words list can be a useful feature to suggest related words but is a lot less relevant than the results and the relationship between them so we preferred to show this list only when needed, by pressing a button for example. Another drawback of this design is that these passage windows use a lot of space so it cannot display many text passages on the screen.

4.2 Final prototype design

The shortcomings of the previous design and the above decisions have lead to the final prototype design shown in Figure 4.2.

![Figure 4.2](image)

**Figure 4.2:** The final prototype design where the user has clicked on the first text passage of the first query. Each view will highlight the information that belong to the clicked passage.

The tool uses a simple way to give the user a quick overview of all the results by viewing them...
in a list. It is best to not lose parts of the tool that the user already likes about it, so we got rid of the passage window design and reverted back to displaying the text passages as a list. We want this results list and the hierarchical book structure to be visible at all times so that the user always knows with which text passages and which books he is dealing with. Therefore, we chose to divide the part of the screen after the input query in three views. The Book Structure View containing the hierarchical book structure on the left, the Text Passage View containing the results list in the center and the Context View on the right to provide more information on the context of a selected text passage. Each view will have its own set of options to manipulate the visualization of that view to include or exclude parts of the corpus in the results UR3. We use the blue highlighting to link the views together. Clicking on a passage would highlight the passage with its corresponding book in the Book Structure View and the corresponding context in the Context View in blue. This context is shown by displaying the original PDF file of the corresponding book where other passages, from the Text Passage View, stated in that same book would be highlighted with a different color. As these text passages can be scattered across the entire book, we came up with the Stripescrollbar that is shown next to the scrollbar of the Context View. Each stripe in the Stripescrollbar represents a text passage and is colored the same as the highlights in the text. This design gives a quick overview of where all highlighted text passages are in the book. Clicking on a stripe would scroll the Context View to that specific passage.

Clicking on a previous query would give the results belonging to that query and changes the three views accordingly. The queries can then be compared to each other with the Compare button that shows their overlap in a new view, the Overlap View(UR7). We measure the overlap between entities (terms, words, phrases, sentences) to be the cosine distance, i.e. the score that is obtained after multiplying the sum vectors, of those entities. An UpSet-like visualization is presented that shows these scores for all query combinations. In Figure 4.3 you can see the scores as vertical bars for the participating queries with the filled circles. This visualization makes it possible to visualize all query combinations in a way that makes it easy to accurately read the data.

Clicking on a previous query would give the results belonging to that query and changes the three views accordingly. The queries can then be compared to each other with the Compare button that shows their overlap in a new view, the Overlap View(UR7). We measure the overlap between entities (terms, words, phrases, sentences) to be the cosine distance, i.e. the score that is obtained after multiplying the sum vectors, of those entities. An UpSet-like visualization is presented that shows these scores for all query combinations. In Figure 4.3 you can see the scores as vertical bars for the participating queries with the filled circles. This visualization makes it possible to visualize all query combinations in a way that makes it easy to accurately read the data.

Figure 4.3: After pressing the Compare button, the user will be shown an UpSet-like overview of the overlap between queries.
CHAPTER 4. DESIGN DECISIONS

4.3 Revised decisions

During the development of BolVis we found that some design decisions we made for our prototype needed to be revised for its implementation. For one, highlighting a specific sentence in an original PDF file has proven to be very difficult and considering the available time we had, we decided not to implement this. This has lead to modifying the priority of requirement UR11 to won’t have. As a result, the Context View became a lot less relevant to show all the time so we removed it from the main screen and in the implementation we now display this in a new tab of the browser. Instead, we created a Favorites View on the main screen to visualize the text passages that are selected for close reading and comparison (UR6).

The Stripescollbar is also not needed in the Context View anymore. Instead, we thought it would still be useful in the Book Structure View when it would act as a small representation of the expansion panel. Even though the expansion panel reduces the vertical space, it is for most computer screens still too large to view entirely. With the addition of the Stripescollbar, a quick overview is provided of all highlights and expanded entries in the Book Structure View. The final implemented design of BolVis can be seen in Figure 4.4 where its functionalities and visualization are further explained in Section 5.4.

![Figure 4.4: The final implemented design. Visualization after entering the queries abfolge, gott, and grund und folge. Currently, the results for gott are shown, several passages are added to the Favorites View, the user has pressed the first text passage, and is hovering over the third passage.](image)

Another revised design decision we had to take was to not show the overlap between queries in a new separate view but combining it with the available views. It makes this information always visible while not losing sight on other information. First we combined the overlap score with the Text Passage View. This score is now visualized as vertical bars to take up the least horizontal space in order to use the rest of the available space for viewing the passage itself. The vertical bars have four quantitative steps to indicate the score difference more effectively. If the score is between 0.75 and 1.0, all four steps are filled, showing a full vertical bar. A score between 0.5 and 0.75 gives a vertical bar with three filled steps, and so on. All scores below 0.125 will result in an empty bar since these are too low to be considered relevant.

We then combined the boolean set representation with the Book Structure View because we wanted to visualize where this overlap occurs, on book level, chapter level, section level and so on (UR7). Each entry in the hierarchical book structure can either contain or not contain one or
more passages from the Text Passage View. We show this by displaying filled or empty circles respectively at each entry of the Book Structure View as this is an accurate and clear way to visualize this boolean representation.

Originally, we thought of using input boxes where the user had to supply two values to set as range. In each view the entries are removed when the corresponding attribute of that entry does not fall into that range. The downside of this is that you have to catch wrong inputs like characters or integers with less or more than four digits. Range sliders solve this problem and are more intuitive. We got this design from [21] since it offers a lot of functionalities and works well with responsive web pages.

Figure 4.5: Using input boxes instead of range sliders.
Chapter 5

Prototype implementation

5.1 Ariadne

OCLC has developed an information exploration tool based on Ariadne\cite{19, 20} that is capable of browsing through millions of entities, using semantic search. Entities are treated as authors, topical terms (words), journals and Dewey\cite{17} decimal classes assigned to journals. The tool starts off with an off-line preprocessing phase and is followed by an online visualization process. The following section will explain these two steps.

5.1.1 Combining off-line and online processes

During the preparation phase, the tool executes a procedure for semantic indexing. The first step in this procedure is that it removes stop words and selects the most frequent terms, that can either be single words or two-word phrases, from the database. It then constructs a semantic matrix representation of all entities. Every entity is represented by a co-occurrence frequencies vector where the row vectors are defined as the context of each entity. The relatedness between different entities is then computed based on their shared context. The semantic similarity between entities is determined by computing the cosine similarity of the context of these entities. A high value for the cosine similarity between entities signifies a high overlap between the contexts of these entities, meaning that there is a high similarity between them. This makes it possible to calculate the similarity between sentences or any strings of words and is instantly applicable on the Bolzano dataset because it works on text that is not annotated. A visual representation of the constructed semantic matrix is given in Figure 5.1 as Matrix $E$. Random Projection \cite{3, 10} is then used to reduce the dimensionality of $E$. The idea of Random Projection is that points in a sufficiently large dimensional vector space can be projected into an adequate lower dimensional vector space that retains the distance between any two points of this space. We can perform this on matrix $E$ by multiplying it with a fixed random matrix $RP$ that contains only 1 and -1. This decreases the number of column vectors to get a low-dimensional semantic matrix $E'$.

The next step in the off-line procedure is constructing a vector representation of each document in the same database. The tool aggregates all entities that belong to one document in the semantic matrix $E'$ to acquire a set of unique vectors. The weighted average of these vectors is now the semantic representation of this document. Each entity in that document is given a weight equal to the inverse document frequency to the third power. This causes frequent entities to have little effect on the final document representation. Since these document representations are in the same semantic space as the entities, it is possible to compute the similarity between entities and documents. The most frequent terms are considered the vocabulary (stored in the vocab file where its corresponding vectors are stored in vector). Similarly, this also applies to the documents that are stored in docvocab with the representing vectors in docvector.

In the online process, terms (a single word, a phrase, or one or more sentences) from a search query are compared to the entities in the low-dimensional semantic matrix $E'$ and to the document
5.1.2 Current interface

The tool has a plain interface and can be found and used at http://thoth.pica.nl/bolzano/relate. In it, the user is able to enter either a specific search query and/or a document number, referring to the ID of a specific sentence in the corpus, as shown in Figure 5.2. It is also possible to enter a certain number of top results the user wishes to see. We can see from the figure that this tool will show 20 results in this case.

![Figure 5.2: The Bolzano tool.](image)

After entering a search query, the tool returns a list of relevant terms followed by a list of available passages as displayed in Figures 5.3a and 5.3b respectively. These works have been digitized in the DjVu, PDF and text formats as discussed in Section 1.2. Each entry in the list of relevant terms has two attribute values. The first is the relevance value of this entry (indicated as a number in \([0,1]\)) and the second is the number of occurrences this word has in the Bolzano
dataset. This list is sorted by the relevance value. Clicking on one of the entries in this list will result in a new search for this entry.

Each of the passages in Figure 5.3b has 2 attribute values that can be found in the first line of each result. This is indicated with a red I in Figure 5.4. The first is a relevance value and the second is a String that contains where this sentence can be found in which book, paragraph, section and sentence. First the book is mentioned followed by the section, paragraph and finally the sentence number. Next to these attribute values the functions context and query are shown. When the user presses context of a passage, it reveals the surrounding passages that matches the clicked context passage by making the non-surrounding passages light gray as we can for example see in Figure 5.4, at III. The context of a passage is thus defined as the surrounding sentences of that passage. The second function query queries the entire sentence in the results lists and treats the entire passage next to it as a query. This is equivalent to cutting and pasting the passage in the search bar.

The second line of Figure 5.4, indicated with a red II, conveys the relevance of this results. Each word in this sequence has a weighted value that indicates its importance. The weights of each of these words are then multiplied with their corresponding vectors from the file vector and then added with each other to calculate a sum vector. This sum vector is then multiplied with each sentence vector from the file docvector to come up with a score for that sentence. So the score you see in line II is this specific score computed for that sentence.

Finally, the third line, indicated with a red III, is the passage itself. As indicated above, this passage can either be presented in a black font or in a lighter gray font to indicate its importance.

Figure 5.3: Lists that the Bolzano currently returns after a search query.
5.2 Preprocessing data

The data that is described in Sections 1.2 and 1.3 is preprocessed by the exploration tool discussed in Section 5.1. We decided to replicate the functionalities of this tool in Java\(^1\) since this programming language is object-oriented, meaning that we are able to create modular applications and reusable code. Our replicated tool first reads and stores the contents of the files vocab and docvocab with their corresponding binary files vector and docvector respectively. Note that originally, the first line of the binary files consists of the number of lines of its corresponding vocabulary file followed by the vector size.

After a search query is entered we compare its terms to vocab. The weights of the matching phrases are multiplied with the corresponding vectors from vector. These multiplications for each phrase are added to each other to obtain the sum vector for this query. This sum vector is then multiplied with each vector in vector to compute the relevance score for every corresponding phrase in vocab. This score is a real number in the interval \([0.0; 1.0]\). We say that a phrase with a higher score is semantically closer to the compared query than a phrase with a lower score. The phrases are then sorted on the highest score where only a number of top results will be shown, specified by the user. A similar argument holds for computing the relevance scores for the text passages. Instead of using vector and vocab we use docvector and docvocab to calculate this score for each text passage.

Our preprocessing tool also reads and stores the contents of bolzano_sentences_20180608.txt, sentences_original_20171118.txt, and tab.txt where the latter is read and stored twice in two different data structures since it is used for retrieving both the hierarchical data on all books and the hierarchical data corresponding to each text passage. The hierarchical structure is the same for each book and is built in the following way, depicted in Figure 5.5. As we can see in Figure 5.5, every book starts with only its ID and its book title. We see an example for the book Briefe Prihonsky where it shows that it has a chapter called Briefe Bernard Bolzanos 1836-1845. This chapter then contains the letters 115, 116, and 117 and so on. This is followed by a different book Zur Physik II where the same rules apply.

For reading and parsing this structure, we have devised an algorithm that reads the contents

\(^1\)https://www.java.com/
CHAPTER 5. PROTOTYPE IMPLEMENTATION

Figure 5.5: An example of what the hierarchical structure of the books looks like, stored in tab.txt.

of tab.txt and output a corresponding JSON\textsuperscript{2} format. It also checks the name of each book, chapter, subchapter, and so on, for a valid year to fill in the year attribute since this information is often not present for a book in the data. Applying this algorithm on Figure 5.5 would for example result in the following JSON format.

\textsuperscript{2}https://www.json.org/
CHAPTER 5. PROTOTYPE IMPLEMENTATION

{"chapters": [
{
"name": "Briefe Prihonsky",
"year": "null",
"sub": [{
"name": "Briefe Bernard Bolzanos 1836-1845",
"year": "1836-1845",
"sub": [
{
"name": "Letter : 115",
"year": "null",
"sub": []
},
{
...
}
]
}
],
{
"name": "Zur Physik II",
"year": "null",
"sub": [
{
"name": "Uiber Warmessung. u.A.",
"year": "null",
"sub": [
{
"name": "Section : 1",
"year": "null",
"sub": []
}
]
}
],
{
"name": "Zur Physik. II.",
"year": "null",
"sub": [
{
"name": "Section : 1",
"year": "null",
"sub": []
}
]
}
],
{
"name": "Bibliographie",
"year": "null",
"sub": []
}
],
{
"name": "Personenregister",
"year": "null",
"sub": []
}
]}
}
5.3 Implementation

In building the BolVis web application we used the Client-Server network architecture, making it easy for users to access it through their browser, and is depicted in Figure 5.6. For the client side we used HTML, CSS and Javascript in combination with the Angular framework\(^3\). This open source web application framework is developed by Google and offers many customizable user interface views with straightforward template syntax. On the server side we store all required data files together with the implementation of the preprocessing tool described in Section 5.2 and the communications handler, where both are written in the programming language Java.

![Figure 5.6: The Client-Server model that is used in our network architecture.](image)

We have divided the functionality of the BolVis application into four different main components. The Angular and Data files components are both independent of the application. In Figure 5.7, we give an overview of this overall architecture and show the dependencies and interactions between the components.

![Figure 5.7: The overall communications architecture of the Bolvis application.](image)

First, the GUI component is written in HTML/CSS and contains the layout of the web page and includes indications for the appearance of the user interface by means of HTML tags. The custom tag attributes are interpreted by Angular as directives to associate elements of the page to a model which is represented by means of variables from the Controller component. They can be manually set or retrieved from static and dynamic JSON data. So when the user interacts with the interface, the GUI only retrieves data from Angular and sends and retrieves information from and to the Controller. It visualizes a representation of the state of the application that is maintained in the Controller.

Second, the Controller component essentially holds all the necessary data to provide the GUI with the correct state of the application and retrieves this via the Communications handler from the Preprocessing tool. The Controller contains different methods written in Javascript that quickly affects the behavior and the contents of the web page.

The Communications handler component, as its name suggests, is responsible for the communication between the Preprocessing tool and the Controller. It calls the necessary methods from

\(^3\)https://angular.io/
the Preprocessing tool for each new query request it gets from the Controller, then structures the retrieved information into an acceptable JSON format where it finally will be sent back to the Controller. The same steps apply when retrieving the hierarchical structure of the books.

Finally, the Preprocessing tool retrieves information from the Data files provided by OCLC and preprocesses this data in the manner that is described in Section 5.2. It has methods for reading and processing both the data files and the query input, and has methods for computing relevance scores of the search queries.

5.4 BolVis

BolVis can be found on the URL: http://glammmap.win.tue.nl:8080/BolvisApp/ and currently works best with the Google Chrome browser. As soon as you open this web page it starts with only two input fields and a Search button as you can see in Figure 5.8. Users kick off by entering a query in the Query field, that can be a single term, a certain phrase, or a specific sentence. Optionally, a number of results can be entered in the Amount field. By default, this number is set to 50.

![Figure 5.8: The starting input view of BolVis.](image)

After entering a search query, the user will get access to the rest of the user interface that consists of three main views. For the remainder of this section, we will discuss each of these views in more detail by means of an example. Let abfolge, gott, and grund und folge be the first, second and third search query we enter respectively, each showing 50 results. A complete overview of the visualization for this example is given in Figure 4.4 of Chapter 4.

5.4.1 Book Structure View

This hierarchical book structure overview contains all books from the Bolzano dataset that are shown in an expandable list. The books are at the top level and beneath it are chapters, subchapters, sections and/or letters. Not all books have the same number of levels. Each entry (book, chapter, section, et cetera) starting with the ">" sign indicates that this entry is expandable. After expansion, this sign changes into "∨", meaning that the entry is expanded. We can see in Figure 5.9b for example that the book RW3 110-166 is expanded where only the first chapter can be expanded while the rest of the chapters cannot.

Each entry also has a number of circles where the first circle corresponds to the first query, the second to the second query and so on. A filled circle indicates that at least one text passage in the results list from its corresponding query occurs in that entry while an empty circle indicates otherwise. Looking at Figure 5.9b again we can see that in our example there are results from the queries gott and grund und folge occurring in the RW3 110-166 book. More specifically, they occur in the chapter Lehrbuch der Religionswissenschaft but do not occur in the other chapters.

This view can be manipulated visually either by sorting the entries or changing the range slider. Initially, the checkbox for sorting is marked and by default it is set on Year as in Figure 5.9a. In the preprocessing phase described in Section 5.2, each entry gets a Year attribute containing the 

---

4Online: July 30th, 2018
5https://www.google.com/chrome/
year or year range the entry is from. The sorting algorithm Merge Sort is used to sort all entries. In Figure 5.9 we show that the entries in the Book Structure can be sorted by Year (Figure 5.9a) or by Relevance (Figure 5.9b). Unfortunately, not all entries have a valid year attribute since the data is incomplete, so those entries are not sorted. Clicking on the radio button Relevance will sort the entries on all levels in the hierarchy by the number of filled circles. The ones with the most filled circles are shown on top at each level and are probably most relevant. This is shown in Figure 5.9b where Wissenschaftslehre is on top of the entire list and lehrbuch der Religionswissenschaft is shown on top within RW 110-166.

The range sliders are used to view only the entries with a year attribute that falls into this range. The sliders are initially set to the minimum and maximum value of the year range in which the books from the Bolzano dataset was published and is automatically computed in the preprocessing phase. Moving the sliders will remove entries from the view as we can notice from the difference between figures 5.9b and 5.9c. Entries Philosophische Schriften 1832-1848 II, Philosophische Tagebcher 1811-1817 I, Erbauungsreden 1813, Philosophische Schriften 1832-1848 I and Philosophische Schriften 1832-1848 III are removed in Figure 5.9c after setting the left slider to 1818 and the right slider to 1846. Putting the sliders back to their original values will make the removed entries reappear.

![Figure 5.9: The Book Structure View after entering three queries. The entries in this structure can either be sorted by the radio buttons or filtered by using the range slider. The Stripescrollbar indicates which entries are expanded and which are highlighted.](image)

5.4.2 Text Passage View

Each tab in the Text Passage View represents a query. Every tab-label has a checkbox, a phrase list button (\(\times\)), a label containing the entered query, and a button to remove the tab (\(\times\)) respectively. A list of suggested phrases that are semantically close to the selected query will be shown when pressing on the corresponding phrase list. We see this for example in Figure 5.10 for the selected query abfolge. Pressing on one of them will replace the selected query and its results by the suggested phrase. The Book Structure View will also be updated accordingly.
CHAPTER 5. PROTOTYPE IMPLEMENTATION

Figure 5.10: BolVis provides a list of suggested queries that are semantically close to the selected query *abfolge*.

Every tab holds its own results list containing text passages and its own option settings. The text passages most closely related to the query are shown on top. Each passage begins with a set of vertical bars, one for each query, followed by the passage number and the text passage, and finally a *directory*, i.e. the exact place this passage is found in the corpus. The vertical bars show the relevance score of that text passage compared to a query. Looking at Figure 5.11a we can see the text passages that belong to the query *abfolge*. The passage colored in dark gray is passage number 13 and has three vertical bars. It shows that this passage has a relevance score between 0.5 and 0.75 for *abfolge*, as expected, but for the other queries it has a score close to 0.0, meaning they are not semantically close.

Hovering over a passage causes the passage to be colored darker gray where it also reveals three buttons. The magnifying glass button on top simply queries this passage, a functionality that the Ariadne-based tool also has (UR14) and described in Section 5.1. The heart button in the middle will add the text passage to the list in the Favorites View. All passages with a black colored heart have been added to this list whereas the ones with an empty heart have not. The context button on the bottom opens a new tab in your browser that shows the context for that text passage. This context is in most cases the original PDF file of the book where the text passage is from. This PDF is loaded and displayed using the library PDF.js [16], a PDF viewer that is supported by Mozilla Labs. Using this library, we were able to directly view the correct page in which the text passage is stated. It also comes with a nice interface and handy functionalities for reading the documents. For other passages, the corresponding txt file will opened in the new tab.

We implemented a line clamp for each passage that truncates its text to a maximum of three lines. It truncates the directory to a maximum of one line. Clicking on a passage will not only highlight it but also remove both line clamps and enlarge the passage so that the whole text fits in it. We see this for text passage 13 in Figure 5.11b. The line clamps will only be added again when the same passage is clicked.

The text passages can be filtered to include only those from selected books and/or sections. BolVis looks into all directories of the passages and creates a list with books to filter from. *Filter books* is a button that shows this list as we can see in Figure 5.12a. Initially, all books are marked and unselecting a book will remove all passages of that book from the results list. In case this list contains many books, as in Figure 5.12a, the user can unselect all books at the same time and re-enable them one by one if necessary. The sections can be filtered by moving either a side of the range slider. We see in Figure 5.12b that only the text passages with sections between 183 and
CHAPTER 5. PROTOTYPE IMPLEMENTATION

(a) Hovering over the text passages will reveal three buttons, each with their own action.

(b) Selecting a text passage will not only highlight it but also reveal the entire text.

Figure 5.11: The Text Passage View that shows the results for *abfolge*.

379 are shown in the results list. Therefore not showing at least passages 1, 3, and 5.

5.4.3 Favorites View

Text passages that have been added to the favorites list will be shown in the Favorites View. This view will only appear when there is at least one passage in the favorites list. Adding the same passage twice will put that passage on top of the list. The passages in this view are visualized in the same manner as in the Text Passage view where they can also be filtered in the same way. The only exception to this is that the hearts in these passages are colored black. Clicking on this black heart will remove the passage from the favorites list. The *Filter books* button and the range slider are updated each time a new passage is added to the list. We can see this in Figure 5.13 where first one passage is added to the favorites list in Figure 5.13a and then a second passage is added in Figure 5.13b. The only additional function this view has is that this favorites list can be exported to CSV by pressing the download button on the top right. Each line in this exported CSV contains the search query, the text passage, the relevance score between the search query and the text passage, the directory, and the corresponding PDF or txt file respectively.
Filter books shows a list of books that occur in the passages. Passages where its corresponding book has a marked checkbox will be shown.

The range slider will filter the results list by showing only the passages where its section or letter falls into this range.

Figure 5.12: Each query has two ways to filter the passages and has its own option settings, meaning that filtering one query does not affect the other.

(a) Favorites View for one passage.  
(b) For two passages.

Figure 5.13: Favorites View for two passages.
Chapter 6

Conclusions

6.1 Evaluation

This project started with a specified goal that resulted in the creation of several general, capability and constraint requirements. All the design decisions are based on meeting those requirements. The tools and libraries we implemented were chosen to make the application look good and make it interactive. In BolVis we can instantly see that the expansion panel of the Book Structure View with its offered options in our design meets requirements UR1 and UR2. The circles in each entry show the similarity among the queries involving the books on chapter and section level (UR7). The Text Passage View contains tabs for each search query (UR5) where in each tab the results can be filtered on books and sections (UR3). Each text passage shows the overlap between queries as vertical bars (UR7) and has three buttons when hovered over where their functionalities meet the requirements UR4, UR6, and UR8 respectively from top button to bottom. The suggestion list that can be requested for each search query also provides a way to perform a search within the context of the first search (UR4). After a user has requested an original PDF or text file for a text passage, the exact page of that passage is shown (UR10). The same functionalities and filters are implemented for the Favorites View so that it allows for comparison and close reading of selected text passages (UR6). In addition, it also has the functionality to download all data from the favorites list into a CSV file for future reference or inspection.

BolVis has been designed to perform as fast as possible while supporting the same functionalities as Ariadne (UR14). In the design phase we tried to make the application as responsive as possible so that it would support devices other than desktop computer or laptop screens as much as possible (UR15). However, we do not guarantee that there would not be any distortions in the interface using other devices. The performance is hard to test whether it actually returns the results within 0.5 seconds in 95% of the time as in UR12 since we do not have benchmarks. In our experience, BolVis is able to find relevant passages for our research very quickly. Using the client-server architecture, it can also support multiple concurrent users (UR13). It however does not support other devices than laptops or desktop computers other languages than German, other corpora than the work of Bernard Bolzano, and other web browsers than Google Chrome as stated in the constraint requirements UR16, UR17, and UR18.

Performing the steps in each of the use cases stated in Section 3.4 will both result in reaching the required goal. In the remainder of this section we will evaluate the usefulness of BolVis for text-based philosophical research by means of answering a research question in the form of a use case [18]. This research question is involved with the notion of one-to-one correspondence in set theory. This notion says that two sets $A$ and $B$ are equal in size when every element $a \in A$ there is one element $b \in B$ and vice versa. We know that Bolzano has rejected this notion as an adequate criterion for size equality for a while but at some turning point came to accept it. We now want to investigate the reasons for this turning point. We start this investigation in the book Wissenschaftslehre paragraph 102 (WL §102) where an argument is stated regarding logic that
Bolzano later rejected. This paragraph contains many technical terms that are nowadays either not generally used anymore or have a different meaning, making it very hard to understand. The title of WL §102 already provides some information: “Keine endliche Menge von Maßen genüget, die Weiten aller Vorstellungen zu messen” meaning “No finite set of measures suffices to measure the width of all ideas”. From WL §48 we can deduce that Bolzano sees the term “ideas” as abstract objects that form the building blocks of propositions that bear falsity and truth and acts as interpretations of sentences. Now we want to use BolVis to aid us in finding what Bolzano exactly means with “to measure” and the “width” of all ideas. We start by querying “Weite” (meaning width) and after a close inspection of the results we find the following text passage in the first section of Größenlehre (GL §1), as shown in Figure 6.1.

“Unter der Weite einer Vorstellung verstehen wir nämlich eine Größe, welche die Menge aller ihr unterstehenden Gegenstände mißt, und wir sagen sonach, daß ein Paar Vorstellungen gleichweit sind, wenn sie gleichviele Gegenstände haben, und daß die eine weiter sei als die andere, wenn jene mehr Gegenstände hat als diese, unangesehen, ob die Gegenstände der Einen auch zugleich Gegenstände der anderen sind oder nicht.”

Figure 6.1: The results list for the query Weite after we use Filter books to show only the results from Größenlehre. This is also highlighted in the Book Structure View.

From the results of the search query we find that with the “width” of an idea Bolzano means the quantity or size (Größe) of the set of objects to which that idea refers. An example in mathematics is that the root of the equation \(x^2 - 5x + 6 = 0\) refers to two objects: 2 and 3, so the width of the idea root of this equation is two.

We continue by querying “Messen” (meaning measure) in BolVis and find in the fourth text passage the exact title of WL §102 and decide to click on the magnifying glass that queries this passage. The eighth text passage is interesting and is also displayed in Figure 6.2:
“Inzwischen ist es doch wahr, daß wir unendliche Mengen als solche nicht messen können; und daraus eben ist ersichtlich, daß man hier unter Weite etwas ganz Anderes als die bloße Menge der unter einer Vorstellung enthaltenen Gegenstände verstehet.”

Figure 6.2: After querying “messen” we can query the entire text passage title of WL §102. BolVis shows an interesting result in blue.

This sentence stated in WL §93 indicates that infinite sets cannot be measured which contradicts with the contents of the body of WL §102 where Bolzano talks about series of infinite sets and their measurements, indicating that they can be measured. Finally, we discover that Bolzano rejects this latter argument of WL §102 in a letter to a student where he says he was wrong to measure the infinite sets. We can thus conclude that one of the reasons for accepting the one-to-one correspondence notion as an adequate criterion for size equality is for rejecting the argument that Bolzano presented in WL §102.

6.2 Conclusion

This thesis presented an interactive visualization tool that helps in text-based research significantly. Its design meets the stated requirements and applying the use cases from Section 3.4 in BolVis results in achieving the required goals. The use case from the previous section shows that our tool can quickly find relevant text passages to obtain a better understanding of the technical terms or phrases presented by Bolzano in his works. The usefulness of our approach lies in the visualization that provides the user with only a part of the corpus and a justification of why it shows the results and in what order. Changing this order by manipulating the visualization is an effective way to look at the results from another angle. Unfortunately, we are not certain that BolVis shows all relevant passages since it does not perform an exact search where it may leave out some passages in our query search. During the analysis we can also not be certain that we always query the correct terms or examine a sufficient number of results. However, these issues also happen during close reading by hand.
6.3 Future work

After finishing our work, there are still some issues that can either be improved or added to the tool for future work. There are the requirements that were not implemented but can still be useful functionalities for future work.

After querying a term, only the relevant text passages are shown in the results list. To see more of the context of this text passage a user now has to click on a button that opens the original PDF or text file in a new window. If the user could already see a few sentences before and after each search result (UR9), it would provide a quicker overview of the context of each text passage. This could be implemented to appear after clicking on a passage so that the overview of these relevant results is maintained.

In opening the original PDF file we did not implement the functionality that highlights specific sentences. However, this can still be a useful feature when a user searches through the context to find the corresponding passages quicker. The library PDFjs we use for viewing the PDF files offers functionality that can be used for this purpose. We now show this PDF file in a new tab but for future functionality it might be better if this is embedded within BolVis. Now BolVis can only pass data to PDFjs but with a better integration, the data could be passed both ways so that BolVis can also retrieve necessary information.

Currently our tool focuses on the corpus of a single author. In the future this can be extended to the works of multiple philosophers. As long as the data is delivered in the same format, our tool should have no problem with processing this data. BolVis could then be modified to also find overlap or semantic similarities in queries between different authors to make it useful for conceptual analysis and close reading across the collections of several philosophers.

The boolean representation of each entry in the Book Structure View now only shows whether or not there is a hit of the corresponding query in the stated book. For future work it might also be useful if this representation can also show the number of hits of each query by means of implementing different circle sizes. The higher the circle, the higher the number of hits for example. This might require some adjustments to the design to show the differences in size properly.
Bibliography


