Linköping University | Department of Science and Technology
Master's thesis, 30 ECTS | Medieteknik
2026 | LiU-ITN-TEK-A–26/021–SE

Lightweight Text Visualization for Close and Distant Reading

– A Web-Based Tool for Document and Corpus Analysis

Webbaserad Textvisualisering för Kort- och Långdistansläsning
– Ett Verktyg för Dokument- och Korpusanalys

Philip Robertsson

Supervisor : Kostiantyn Kucher
Examiner : Andreas Kerren

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

Analysis of larger corpora or longer documents can be time-consuming and given current tools for it, difficult for many users. Document and corpus analysis in this context refers to the process of providing users with both close and distant reading representations of individual documents and corpora. This thesis explores the design and development of a text analysis tool with close and distant reading representations and with the goal doing this with lightweight approach and with generalized language support. Additionally, the target user base for this thesis has been widened to encompass more than people with deep understanding of text visualization, done to address the slim scope of earlier tools.

To accomplish the goal of the thesis and answer its research questions, several objectives were defined, and the work was split into implementation and evaluation phases. This split was done with an iterative process in mind, where an initial prototype was tested and evaluated, with a second prototype implemented based on the evaluation, and so on for further prototypes. To limit the thesis work, however, the objectives were defined to only encompass a first, second, and potential final prototype.

The tool was developed as a web-based tool with *JavaScript*, relying on *D3.js* for the drawing of visual elements. The tool was developed with a *component*-based approach where each view visualized different data about singular documents within a corpus or the entire corpus itself. Within this interface was the close reading view which wrote out the text of the selected document in plain text, with possibility for word highlighting and selection from the user. In addition to this were views for word frequencies, showing the number of occurrences each word had in the document, and word trends, showing how significant specific words were in specific segments of the document. A provided corpus could also be visualized through a card-based corpus overview, and the more data-centric node view, representing how similar documents were in the corpus. For further explanation on the similarity between documents, a similarity overview was also added, accessible on clicks on the node view edges.

The evaluation process, on the other hand, was done through result collection on different tests, both with and without users. The user-free tests aimed to find results on the inner workings of the tool and was conducted during the implementation phase to find results which were not part of the user tests and to find possible faults to fix during the user tests. The user tests were instead conducted to find results related to the usability of the tool, both measured and perceived by the participants. These test phases gave results on the processing times within the tool, showing which *NLP* methods were preferable from a performance perspective, and how the different methods represented the data. Additionally, the results highlighted strengths and weaknesses within the tool given the quantitative and qualitative data from the user tests.

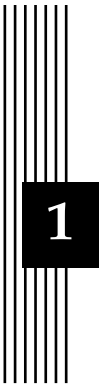
The reasons behind these results were evaluated and discussed implementing a second prototype addressing shortcomings of the first prototype. This second prototype, the discussion leaning up to it and the thesis results was then compared to the Objectives of the thesis, along with being used to answer the proposed Research Questions. Lastly, with the differences between the second prototype and the discussed improvements, a direction for the tool in the future was defined. This also covered uncompleted Objectives and answers to the Research Questions which were deemed to be less direct compared to other answers.

Acknowledgments

I would like to begin with acknowledging the great help my supervisor Kostiantyn Kucher has provided during this thesis. This gratitude is extended both to the academic support he has provided and to the several hours of meetings we have had during these months when several new ideas for the tool was brought to light, which I thank him for immensely. I would also like to thank my examiner Andreas Kerren who has show keen interest in my work and provided me with valuable feedback on the tool, but most importantly this thesis report. Special thanks go out to my partner Linnéa Bergström, who recommended several works of literature to me, which I used while testing the tool, and to my family for supporting me throughout this thesis. Finally, I extend my thanks to all participants who took part in the user tests and provided invaluable feedback on how to improve the tool.

Contents

List of Figures



1 Introduction

To get an understanding of individual texts or books a reader can usually read through said work to get the required knowledge about it. For other cases however where larger collections of works need to be analysed there might arise a problem where the reader cannot read the amount of information required to get sufficient knowledge in a suitable time frame. To circumvent this problem, or help a user to at least get a sufficient overview of the provided corpus a text visualization tool could be utilized [1].

1.1 Background

The field of text visualization focuses on providing a user or viewer with interactive tools to explore a given text or corpus. This can be through graphs, trees, or maps to visualize the information in a more abstract way compared to the traditional method of only writing text with words [2]. It is also important to differentiate between the two reading methods relevant to the thesis, close reading refers to how texts and books usually are read, i.e., word for word on printed or written material. Distant reading on the other hand, refers to a reader gaining information about a given text or corpus by only viewing an abstraction of said text or corpora [2]. The mentioned tool in the introductory sentences is thus meant to provide a user with both methods.

1.2 Motivation

Existing implementations of text visualization tools, such as *Voyant Tools* [3] or the *DoSVis* tool implemented by the *ISOVIS* group [4], provides a user with both close and distant reading visualizations, but these are meant for specific situations that either are too complex for general users, or only covers small datasets. There is therefore a need for a tool which can both write out any provided text or corpora to a reader for close reading and abstract this text to a more general shape or representation for distant reading. This does also necessitate for evaluation on the usability, since the tool should be usable for a general user base.

1.3 Aim

The aim of this master thesis, and the work included in it, is to explore how a *lightweight* web-based text visualization tool can be designed to provide a user with generalizable close and distant reading methods for both singular and several documents with very few assumptions about the text domain and genre (hence the "lightweight" approach). This should be generalised in the sense that multiple languages are supported (at least left-to-right, Latin-based alphabets) and it should be fully developed with front end solutions, avoiding any handling of user data. In addition to this there is an aim to explore how this tool can be generalized to facilitate for a more general user base. This is, as described in Section 1.2, a problem in earlier implementation, here the *general user base* refers to a wider scope of users of different ages, educational backgrounds, and familiarity of visualization tools. The addition of these constraints is aimed to improve the usability evaluation, making it easier to figure out what groups the tool works for and to find measures in to facilitate it for more users.

1.4 Research questions

Some research questions were formulated to aid in the research related to the implementation. These mainly focus on qualitative data since the design of the tool will be evaluated through user input, but quantitative methods are also relevant since performance data will be evaluated. For this thesis project, the research questions were formulated accordingly:

RQ1 What visualization methods are best suited to show a user where specific information can be found within a given text document or corpus?

RQ2 Which natural language processing methods are best suited for lightweight text analysis?

RQ3 Should distant reading views be presented to the user initially alongside the close reading view, or are they more useful if provided on demand only?

1.5 Objectives

Based on the research questions presented in Section 1.4, several objectives have been formulated which aim to highlight specific parts of the thesis work. These relate to the planning, design, implementation, evaluation, and writing processes, where the implementation and evaluation is closely tied to all research questions. The aim of the objectives is to clearly present how the work will be divided, along with formulation of specific milestones. To clearly connect the objectives to the relevant research questions, they are listed after said objective. Objectives for the thesis work were formulated accordingly:

O1 Completion of initial thesis and design plan, **RQ1**, **RQ3**.

O2 Acquisition of relevant text data for lightweight text analysis, **RQ2**.

O3 Validation of different NLP methods for lightweight text analysis, **RQ2**.

O4 Initial design and implementation of user interface and text visualization, **RQ1**, **RQ3**.

O5 First set of user tests to evaluate the initial design and implementation, **RQ1**, **RQ3**.

O6 Design and implementation revisions based on evaluation of **O4**, **RQ1**, **RQ3**.

O7 Second set of user tests followed by small design and implementation changes if needed, **RQ1**, **RQ3**.

O8 Final set of user tests to evaluate the finished visualization tool, **RQ1**, **RQ3**.

O9 Completion of written report paired with work presentation describing the thesis work, **RQ1**–**RQ3**.

As previously mentioned, these objectives are also accompanied by milestones which aim to define clear points of the project where a certain group of objectives should be completed. These were formulated accordingly:

M1 Completion of first prototype, **O2**–**O4**.

M2 Half-time check up on thesis progression, **O5**–**O6**.

M3 Completion of final implementation along with evaluation, **O7**–**O9**.

1.6 Delimitations

Since one of the aims of the implementation is that the tool must be able to run on the front-end of a website there is no possibility to analyse a provided text or corpora with server-side implementations. This means that a major delimitation to the thesis work is that *D3.js* and *JavaScript* should be utilized, but these frameworks still contain many tools which can be beneficial to the work. The use of the *D3.js* library, that will provide the tools for creating views, also requires web-users to have *JavaScript* enabled in their browser, and thus it generally lacks support for browser released before 2010 [5]. As mentioned in Section 1.3, there will also be a limitation on the writing systems supported by the tool, to ensure that the NLP implementation does not overshadow implementation of different views. This does lessen the technical requirements, while still retaining support for right-to-left languages using the Latin alphabet. Based on the user base described in Section 1.3, there is also a limitation on which users can partake in the tests. It is a necessity that a test participant at least has some computer familiarity and knowledge about standard website navigational tools, such as buttons and scroll bars.

2 Theory

This thesis is based on theory in fields of natural language processing, information visualization, and human-computer interaction. As described in the introduction, this also builds on the work done by the *ISOVIS* group on the Document Stance Visualization.

2.1 Natural language processing

In general terms, natural language processing, or *NLP* refers to a collection of methods and techniques used for automated linguistic analysis [6]. This umbrella term does relate both to statistical methods such as basic keyword search, word frequency analysis, and highlighting methods such as n-grams [7, 8, 9]. However, it does also include to machine learning-based approaches such as *SVM*- and *LR*-based classifiers [4]. The aim of these methods is usually focused on translation, information retrieval and extraction, or summarization, meaning that some aspects of it are of key importance to this thesis [6]. This form of computational linguistics is based on general linguistic theory, aiming to encode linguistic knowledge as technical applications [10].

Apart from the n-grams method, another method to evaluate the relevance of singular words within a given part of a text or in a corpus, is with the *TF-IDF* method. This approach utilizes the term frequency, i.e., the number of times the word appears in the sample (document or part of document), and the inverse document frequency, i.e., how frequent the word is in the context (corpus or document) [11]. In addition to these word-relevance evaluation methods, it is also possible to compare how similar two documents within the corpus are to one another. One method is to utilize the previously mentioned *TF-IDF* method to find the relevance of each word in the corpus to each document, creating a vector for each document. These vectors can then be compared utilizing *cosine similarity* to evaluate how similar they are on a scale from 0–1 [12]. Another method would be *Jaccard similarity* where the number of shared words between two documents, A and B, are divided by the total number of words in the comparison. In more direct terms, this method divides the length of the intersection between documents A and B with the length of the union for the same pair, resulting in a value between 0–1 [13].

In addition to this it is also important to note how a given method should be evaluated to improve it. As stated in Section 1.5, the evaluation of different *NLP* methods will be based on results from relevant text data. This generally refers to any data which contains mean-

ingful words that can be extracted, for example *Twitter* posts [7], web pages [14], or books [2]. For methods relying on machine learning, however, it would be needed to also have labelled data for evaluation, since that would utilize a set of input representations and output labels to produce consistent statistical results [15]. Evaluation of the simpler *NLP* methods, such as keyword search, has instead been proposed by Coffman and Weaver to assess the relevance of the results [16]. The relevance of content (a given word or whole text) is in this case decided by the probability ranking principle, or *PRP*, which can be used to figure out the probability that the desired content can be found within a given text [17]. Given the results from the highlighting n-grams method, it is also possible to evaluate the performance of the method with *PRP*. This information can then be used to evaluate the validity of the method, aiming to answer if presents the correct results [15]. It is however important to also keep the questions about reliability and significance in mind as these aims to answer if the results can be replicated consistently, and if differences in the results are based on chance or not [15].

2.2 Information visualization

The field of information visualization, or *InfoVis* relates to how information can be reformatted, allowing users to form a mental representation of said information. In the book *Information Visualization An Introduction* by Robert Spence, he describes that this field does not necessarily involve any digital representations, since static visualizations are possible to print [18]. With the help of digital representations, it is however possible to create more dynamic or interactive visualizations, defined by Spence as visualization tools [18]. There are several different ways that raw data, in this case text, can be represented in an abstracted or reformatted form. Spence does highlight methods like scatter plots, and parallel coordinates which could be used to visualize spatial, and numerical data respectively based on the keyword search results [18].

Compared to the methods described in Section 2.1, evaluation of visualizations is instead mostly done with test participants through qualitative and quantitative methods [19]. The quantitative evaluation is the simpler part of this process since it involves data about how long it takes for a user to complete certain tasks or the error rate while using a given tool. Elmqvist and Yi thus proposes that more emphasis should be put on qualitative user studies when evaluating visualizations [19]. In addition to this, they also describe several different methods, referred to as *patterns*, which can be used both for quantitative and qualitative evaluation. These are based on earlier works by the writers, being described as either arbitrary or subjective, and are divided into the five following categories; exploration, control, generalization, validation, and presentation [19]. Narrowing down the evaluation process specifically to information visualization, however, does present challenges related to human-computer interaction (*HCI*), perceptual psychology, and cognitive reasoning, as stated by Carpendale [20]. When developing tools within the field of information visualization it is therefore crucial to keep these challenges in mind when designing, conducting, and evaluating user tests. There are several ways which this can be achieved, for one it can be done through specific questions aimed to solve the psychological challenges. Also accounting for the different complexities in tasks for cognitive challenges, and acknowledging that previous experiences can skew the results of *HCI* tests, if two interactions are compared to one another [20].

2.3 Text visualization

InfoVis is a broad term encompassing many ways to represent data of several sources, as with examples presented by Spence [18] for example. What this thesis focuses on is more specifically visualization of textual data, i.e., the process of rearranging information found in texts or corpora in a manner that aids with analysis of said information [3]. An example of this would be the implementation presented by Sinclair and Rockwell, *Voyant tools*.

Figure 2.1: Example of *Voyant Tools*.

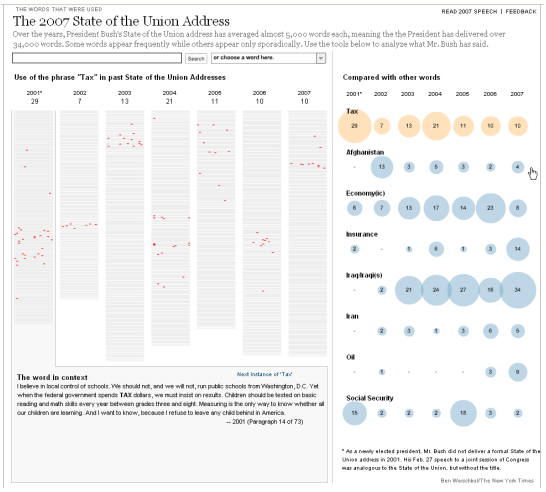


Figure 2.2: *2007 State of the Union Address* visualization from New York Times [22].

Figure 2.3: Example of *DosVis*.

This example features a *cirrus* (also known as word cloud), a graph for word trends, and a close reading representation [3]. They also highlight methods used for the *2007 State of the Union Address* created by New York Times, seen in Figure 2.3 and *Ngram Viewer* created by Google [21][3]. As can be seen in those examples, however, lack any close reading views. Other examples would for example be *Lariat* which focuses on posts on *Twitter*, presented by Nan-Chen *et al.* [7], or the *DosVis* tool mentioned in Section 1.2.

Given the descriptions presented in Section 2.1, text visualization is in this case dependent on *NLP*s to extract information from a given text or corpus. In this context, relevant *NLP* methods would be to find frequencies of words in each text by recording their occurrences throughout the corpus [8]. To get a clearer understanding of the significance a given word has in a text, however, the n-grams method presented by Cohen, or the *TF-IDF* method are applicable [9, 11]. Additional meaning behind certain words could be found through machine learning, performing sentiment analysis to find the "attitude" the writer had, information

that can be passed on to a user through text visualization [4]. In short, however, *NLP*s provides the data needed to generate views in an automated process, thus removing the need for a user to provide additional input to the system.

2.4 Human-computer interaction and evaluation concerns

Expanding on the computer-aided information visualization described in Section 2.2, interaction between the viewer and the visualisation can be a beneficial to extend how much data the viewer has access to. This interaction is defined as human-computer interaction, or *HCI*, while the experimental part aims to perform experiments to evaluate different *HCI* designs. According to Helen C. Purchase in her book Experimental Human-Computer Interaction, *HCI* can be categorised by the following ideas [23];

1. A type of interaction device associated with new hardware.
2. A method of perception using devices that target particular senses.
3. A method for system interaction usually embedded within other software.
4. An interactive system designed to support a complex task.

The focus of this thesis, in regard to *HCI*, will be on points (2–4) since they involve how data can be visualized and interacted with, as well as how an interactive system can be designed to support complex tasks [23]. Visualization of data can range from choices of different graphs or charts, colour encoding, or if text is needed for the viewer to understand the data, all relating to the method of perception [23]. These different visualization parts include design choices which is dependent on the present data and results from user studies. Interaction in a web-based tool include standard browser methods such as scroll bars and the option to zoom in on a given page. Other ideas such as details-on-demand to give the viewer more information when they need it, dynamic queries to allow the viewer to limit the visualization to a specific area of the data, or utilisation of a menu system to ease navigation between several views for example are also methods also to let the viewer interact with the visualisation and focus on parts they desire.

Evaluation of the design of interactive elements, as briefly mentioned in Section 2.2, does include some challenges regarding comparisons between solutions. For example, if a test participant is familiar with solutions present in one tool, there is a risk that said solutions become trivial to the user, thus skewing the results [20]. This challenge thus becomes a question about the validity of the evaluation, where a solution might be wrongfully decided to be usable since the test participant has ease in using it, but this would be a *type I error* [20]. Here it is also important to note what is meant by *usable*. This term refers to evaluation on the *usability*, which in and of itself is split into measurements of effectiveness, efficiency, and satisfaction [24]. The aim of these measurements is to firstly, evaluate how accurate and to what degree of completeness a user achieves a given task, secondly, what the relation between the accuracy and completeness is, and thirdly, how they felt using the solution [25]. Lastly, the evaluation process should handle the question about reliability, referring to how repeatable a given form of measurement is. This is an especially important factor to have in mind when evaluating user test results, since humans are prone to making errors [26].

3 Method

The methodology behind the thesis was based on the previously mentioned theory in Chapter 2, and the objectives described in Section 1.5 were utilized to construct how the work was conducted. The work process consisted of three different processes, the initial design, implementation, and evaluation. The parts described in these processes are mostly presented in chronological order, i.e., the initial design was done before implementation of the tool started, which was then followed by overall evaluation. It should, however, be noted, according to the objectives, that there was evaluation done during implementation as well, this will, however, be expanded on in the relevant Sections. The first part of the thesis work, defined as **O1** in Section 1.5, consisted of planning and constructing a framework for the proposed visualization tool. This planning process utilized ideas from Tamara Munzner's nested design model, seen in Figure 3.1 [27].

3.1 Domain problem characterization

This characterization step, i.e., the first step in this design phase was to figure out the target domain of the tool, this referred to both what tasks the tool should provide, along with what users it was aimed for [27]. The user base which the tool is targeted at has been briefly mentioned Sections 1.2 and 1.3, but to expand on it, a group of target users was defined more clearly. The target domain of the tool was, as stated in Section 1.3 a generalized lightweight text visualization tool that could handle several documents and languages. This mean that sources like books, research papers, economic reports, or text posts on the internet was all meant to be accounted for, which in turned widened the aimed for user base. When the domain had been characterized, it was possible to define the target audience as young adults,

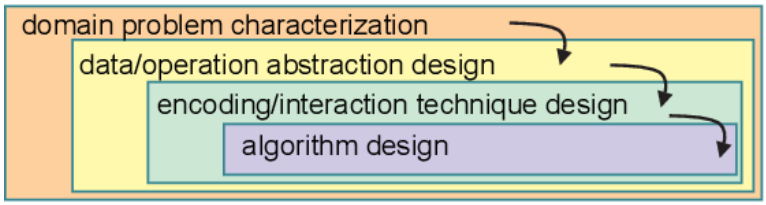


Figure 3.1: The visualization design and validation model presented by Munzner.

mostly in the academic sphere, to working age adults in fields that handle a large number of documents.

3.2 Data and operation abstraction design

As a continuation of the previous layer, there was the identification process which aimed to abstract the low-level domain description into general data types and operations that later could be used to define a high-level abstraction of the tool [27]. Wehrend and Lewis has proposed several relevant objectives for the user, which was applicable in this abstraction step, one being *locate*, which in the text analysis scope, referred to finding where specific keywords were located within a given document or corpus.[28]. Another goal was to ensure that the user could *distinguish* between separate keywords and their occurrences, along with the possibility to *cluster* documents that shared specific keywords. Lastly, it was aimed to facilitate the user with methods to *rank*, *compare*, *associate*, and *correlate* different keywords and documents across the corpus to allow for deeper analysis [29]. Along with the objectives, the general structure of the data was defined based on the previous description of different text sources. To generalize the file processing, it was aimed to only handle plain text, along with putting a restriction on the language support and writing systems, as mentioned in Section 1.6. Where plain text, was regarded as being represented by *UTF-8* characters written out in one sequence, i.e., a 1-dimensional data type [3, 30]. Based on this, a given user was meant to be able to complete tasks related to location-specific keywords, since they could be marked as positions in the document, while also providing the user with some form of distinction through highlights of said positions in each document [30]. Given that there was a goal to facilitate for additional objectives, however, the plain text format had to be expanded to other data types as well. When an entire corpus was utilized instead, which was planned according to Section 3, it would be a 2-dimensional data type since each character position would be tied to a specific document. This was meant to allow for further distinction between separate documents, along with clustering where the user could group together documents that contained the same keywords. Based on the plain text corpora it would also be possible to generate other data types which facilitated for ranking, comparison, association, or correlation. An example would be the formulation of word frequencies based on keyword searches [7], or presentation of word trends across the corpus [3], both created with 2-dimensional data. An additional aim was to connect documents containing similar keywords as networks [30], a way to create a new data type based on the corpus.

3.3 Encoding and interaction technique design

The definition of objectives and data abstraction was then used with the final two levels of the nested model to define the design of the encoding and interaction techniques, along with algorithm design. [27]. As previously defined, the raw data, used to generate different views, was meant to be represented in *UTF-8* plain text, this thus put a demand on the design of the encoding to define what file formats the tool should support. A goal was to support formats such as *.pdf*, *.docx*, or *.odt*, provided by the user as input. Since these are binary formats, however, there was a requirement to decode them if the plain text where to be extracted [3]. Given that this aimed to convert certain file formats to plain text, it was considered as encoding for the close reading representation, since it would become written text, as stated in Section 1.1. Distant reading representations did, however, need different types of encoding, meant to visualize the data in clearer but abstracted ways for the user. Returning to the word frequency representation, the mapping of words paired with the number of occurrences was aimed to be encoded as a vertical bar chart [7, 31] Another approach considered was however to scale present words based on their frequencies, either through a word cloud [3] or a tree map [32]. The goal of these approaches would be to allow for distinction between keywords

along support for ranking and comparison between their occurrences [29], which meant that it would be encoding designs within the scope of the tool. Certain objectives were however not covered by these approaches, such as clustering, association, and correlation, this design challenge was accounted for through the consideration of a word trends view, since it generally visualized where certain words could be found within a document or corpus [7]. The consideration of a network representation was also suitable, given the objectives, meant to allow for creation of node graphs where each node represented a document within the corpus, and its links presenting either how similar two documents are, or which documents contain shared words [33].

Along with the encoding techniques, interaction techniques were also designed based on what the views aimed to present and what objectives the user was meant to utilize. A general technique that was considered to be advantageous with all encoding methods was filtering, i.e., to let the user either show only desired keywords and texts, or threshold them based on the number of occurrences. This was meant to make it easier for the user to focus on desired areas of the data, as an aid to improve their ability to rank, compare, associate and correlate between desired objects [30]. Another interaction technique considered was that the user should be able to click on specific words to highlight them in the close reading view, to allow the user to relate between different views. A similar approach was also considered for a node graph, where clicking on a specific node would highlight it while also showing the corresponding document in the close reading view. Allowing for this was meant to make it easier for the user to find relationships between specific words and texts [30]. Apart from this, there was the question about how views could be organized, i.e., if they should be always shown or if some views should be presented on demand. To take **RQ3** in Section 1.4 into consideration, it was aimed to include both alternatives in the implementation and evaluation phases. Presentation of all views simultaneously would provide the user with a larger overview of all available views in the tool, thus lessening the requirement on user input [30]. Given the number of views proposed, however, there was one problem that needed to be accounted for which related to the available display size. This therefore meant that information could be lost to the user given that the views would generally be smaller compared to other approaches [18]. The other approaches would therefore be to provide some views as *main views*, for example the close reading representation and a word frequency chart. The remaining views, like the word trends and corpus node graph would then be available through menus or tabs, similar to the solution implemented by Sinclair and Rockwell [3].

3.4 Algorithm design

The last stage in the design process related to the algorithm design, which handled to how the encoding and interaction, described in Section 3.3, will be processed automatically [27]. The first thing to be considered is how automatic conversion between binary formats to plain text with *UTF-8* characters, mentioned in Section 3.3, will be done. Since *JavaScript* supports file reading of *.txt* files it would not need to be consideration taken when reading such files with *FileReader*. Similar steps would be possible with *DOMParser* to read *.html* and *.xml* file contents, just with the need to filter out different tags to extract the text content. For other formats, however, such as *.docx*, *.odt*, *.epub*, and *.pdf*, other considerations had to be taken. One approach would be to utilize *Python* libraries such as *Pandas*, *Beautiful Soup*, or *Pdfminer* for the text extraction [34]. According to Section 1.6 however, it would require *Python* to be utilized along with *JavaScript*, which possibly could be done on the front end utilizing *Brython* [35]. Alternatives written in *JavaScript* was also considered, such as *PDF.js* for the *.pdf* format, *officeParser* for *.docx* and *.odt* formats, though with a dependency on *Node.js*, and *epub-parser* for the *.epub* format.

With the plain text handled and raw data available, there came the question of how different views would be generated based on said data. The close reading view was considered

to be written out in a scrollable text box, a solution that was present both in *Voyant tools* and *DoSVis* [3, 4]. An algorithm to handle automatic keyword extraction did also need to be considered as well. The ideas were based on the underlying theory presented in Section 2.1, where utilization of the n-grams method was seen as advantages, primarily since it is language-independent [9]. With this, the goal would be to let the algorithm scan through a given text, with its punctuations and numbers filtered out, symbol by symbol to record the counts of unique n-grams present in it. The counts would then be used to generate scores for each n-gram and based on these scores it would be possible to count how many significant n-grams are present in each word to score said word [9]. In addition to giving a list of scores for each word, this algorithm would also find every word present in the text, thus making it possible to record the frequency of every word in the text, one approach to the word frequency chart mentioned in Section 3.3. If the list of words were to be sorted based on said scores, it was also considered that it could be utilized to find what words are most significant to, thus allowing for comparison between different texts based on shared words and their document-specific significance [9]. What it also allows for is to sort words based on the scores of the n-grams they contain, meaning that it would be possible to find which words are most significant for specific texts [9]. The sorted lists could then be used for generation of the word trends view, highlighting how significances change through the corpus, while also being applicable as links in a node graph, visualizing the most significant words shared between documents.

Finally, regarding interaction techniques, for the filtering methods it was considered that dynamic queries could be utilized, giving the user the possibility to both filter the data based on text input, i.e., searching for specific words, and range sliders, allowing the user to specify a word occurrence or significance range [30]. It was figured that this filtering idea could be applicable to all views since it would remove certain keywords from the word frequency and word trend views. For the node graph, it would instead be used to disconnect nodes which only shared undesired keywords, while texts which contained keywords desired by the user, would remain connected, similarly to the node graphs found in the *SocialAction* visualization [33].

3.5 Design evaluation

Lastly, it should be noted that the nested model was meant to be used iteratively, i.e, proposed ideas throughout this Chapter were subject to change, based on evaluation. Lastly, it is however important to note that the nested model is designed to be iterative. The goal was to evaluate the design during both the implementation and evaluation phases of the work, based on four different problems that related to validity. As stated by Munzner, there was a risk that the wrong problem had been identified, a wrong abstraction method had been used, the encoding and interactions did not work as expected, or that the algorithm was too slow [27]. Thus, the implemented prototypes and their designs, **O4–O5** and **O6–O8** (see Section 1.5), were not the only parts subjected to evaluation, but also the design framework, which was planned to be evaluated continuously. Some of the techniques were planned to be validated through quantitative methods, since they would for example focus on answering if the data could be processed in a suitable time frame. According to Munzner, this would be classed as an algorithmic threat, a threat that also covered if the algorithm produced the correct results [27]. Here, correct results referred to if conversion between specific file formats and plain text would extract the text in the correct manner, along with, if the word extraction from the plain text recorded all present words and if the raw data was encoded correctly. The outer parts of the design model were instead planned to be evaluated through user tests, both with qualitative and quantitative methods. Based on ideas presented by Carpendale, these tests were considered to focus on the users' feelings about using the tool, along with how fast the user could complete certain tasks and to what degree of accuracy [20].

3.6 Implementation and evaluation

Based on planned encoding and interaction technique, along with possible algorithms, a visualization tool was implemented in accordance with Objectives **O2**–**O4** and **O6**, while also handling some user free testing. This phase was also conducted with the delimitations related implementation from Section 1.6 in mind. The evaluation phase of the thesis was based on the ideas presented in Section 3.5 and in accordance with Objectives **O5** and **O7**–**O8**. Thus, this handled both user tests, but also the aforementioned user free testing.

4 Implementation

As indicated by the descriptions of the implementation focused objectives mentioned in Section 3.6, they also include planned design goals, i.e., initial mock-ups which gave a structure to the implementation phase, such as simple layouts or ideas for possible views. The first part, **O2**, of this process focused on acquisition of useful text data and different documents that could be used to test the algorithms. These were, as described in Section 3.1, represented as different text formats, i.e., books, research papers, economic reports, all which handled different subjects. What this was meant to be used for was to evaluate if the tool covered a general scope, which was further accounted for by having different languages represented in the data, as stated in Section 1.3. The tool itself was implemented as a front end website, where *HTML* and *CSS* was used as a foundation, while necessary functionality and the interactive parts of the user interface was created with *JavaScript*, and *D3.js* as stated in Sections 1.6 and 3.4.

4.1 Corpus acquisition

As described in Section 1.3, there was an aim to implement a tool with multi-language support; thus it was chosen to create both English and Swedish corpora, based on the author's language proficiency. These corpora contained 58 English documents and 141 Swedish documents, which as described in Sections 3.1, 3.3, and 3.4 was from different sources, with different subject matters, and in different formats. It is important to note here that the large difference in file count was caused by including the entire poem collection *Fredman's Epistles* by Carl Michael Bellman as 82 separate *.html* files. Additionally, these were the only *.html* files present and was meant to show how well text extraction worked with the proposed *DOMParser* method considered in Section 3.4. To avoid any possible copyright concerns, it was also chosen to only include documents in either public domain or open access. For the Bellman poems, they were collected from the website *Bellman.net* created by Thord Lindé, were they are collected as separate pages, thus with the ability to save them as *.html* files [36].

Swedish literary sources, such as novels, plays, and more recent poems, represented in *.epub* and *.pdf* formats were collected from the digital library *Litteraturbanken*, which makes literary work that is deemed significant to the Swedish cultural heritage freely available to readers, researchers, teachers, and students [37]. As per the goal to also include *.txt* files, the Swedish corpus also included a collection of different works available from *Project Runeberg*,

which creates free electronic editions of classic Nordic literature [38]. Since there was an aim to also include economic reports, as described in Section 3.1, there was several *.pdf* files collected from *Finansinspektionen*, the Swedish government agency for financial regulation, covering different reports and financial research [39].

For the English corpus, the collection process was more concise, in the sense that there were fewer sources. Novels, plays, and novellas which had entered the public domain were collected in *.txt* and *.epub* formats from the digital library *Project Gutenberg*, which as with previously mentioned digital libraries, collects and makes thousands of *eBooks* freely available [40]. A similar approach as found in the Swedish corpus regarding economic reports were also taken for the English corpus. In this case, the economic reports were collected from the *Office for National Statistics*, which reports its statistics through the *UK Statistics Authority* to the *UK Parliament* [41]. These reports were represented by *.pdf* files, and, for example, covered yearly stock reports, quarterly *GDP* reports, balance sheets, and profitability reports.

The final source, covering both corpora, was regarding collection of research papers, in this case only represented by *.pdf* files. These were collected from the online archive *DiVA portal*, which published academic literature digitally [42]. For both Swedish and English language texts, the collected research papers mostly covered subjects related to the field of visualization, but also other computer science related research. As already stated, the language choices were done according to the author's proficiency in Swedish and English, but other languages were also considered for internal evaluation, such as German and Italian to cover other Latin alphabet-based languages. Additionally, there was a consideration to also test the tool with Ukrainian for example, to cover other alphabets as well.

4.2 Natural language processing

The implementation of different *NLP* methods, covered by objective **O3** in Section 1.5, focused primarily on a basic keyword frequency count, and different scoring methods for the present words. As mentioned in Section 3.4, plain text was considered the input to the *NLP* algorithms, where raw data referred to filtered plain text, only containing letters and singular spaces between each word. The filtering process was done in a relatively "manual" manner, i.e., with separate `replace()` function calls with different regular expression, or *regex* patterns. This targeted line breaks, *HTML* and *XML* tags, numbers, punctuation and other types of non-letter characters, various forms of hyphens, bullet points from lists, and apostrophes. This filtering method also fixed the spacing between words and ensured that all letters were saved as lower case letters, to avoid the possibility that "visual" and "Visual" would be considered different words for example. Another *regex* approach was also considered, wherein everything but letters would simply be replaced by a space, for example with the `\w` meta-character and filter out the numbers. This would however not include Latin-based characters with diacritical marks, such as ´´Å", "Ä", and "Ö" used in Swedish. It was also tested to utilize the *Unicode character class* meta-character `\p`, with relevant alphabets defined to both remove the possibility of missed non-letter characters (given the manual setup) and to further generalize the language support of the tool. In the case with Latin-based alphabets (including letters with diacritical marks) the method simply utilized the regular expression `/\P{Script_Extensions=Latin}+/gu`, where the meta-character `P` marked all characters that were not part of the Latin Unicode character class, and `gu` defined that the matching process was global for all Unicode characters [43, 44]. This approach did, however, present the problem with shortened English words such as "I'll" or "We'll", which became to the four words "I", "ll", "We", and "ll". To circumvent this, it was chosen to first remove apostrophes completely, without a space, in this case turning the examples into the two words "Ill" and "Well", which of course were different words compared to the initial ones. It was also considered handling this through conversion to the full form of the words; thus the example words here would instead be correctly identified as "I will" and "We will", but since this would

require a large list of shortened words and their full forms it was chosen to use the simpler approach, according to the aim of having a generalized tool as described in Section 1.3.

With the plain text processed to raw data, each document was split into an array, done according to the singular spaces present after the filtering process, to only have one word per array position. Based on the content within, the word array was identical to the raw data, but it proved to be a suitable way to record the counts of the present words in each document. Since this only saved each unique word paired with its count, it meant that the original text order was removed and thus the plain text and raw data was thus kept to later generate the close reading view. Moving on to the *NLP* methods used to generate the different views, there is firstly the *basic word count*, briefly mentioned in the preceding sentences. This was initially done through a nested loop algorithm which went through two arrays, one being the input array of present words and one being a newly created array meant to store all unique words along with the number of their occurrences. However, after taking the algorithmic problem related to speed, as proposed by Munzner, into account it was instead chosen to first sort the input array alphabetically, and then processing it in one loop [27]. This generalized function was thus able to count the unique words present in the entire corpus, an entire document, or in individual segments of a given document, all depending on what data was required for generation of a specific view.

The other *NLP* method considered in Section 3.4, was a similar approach to the n-grams method proposed by Cohen [9]. However, based on research question **RQ2** it was also chosen to implement the *TF-IDF* method to compare the generated word score of the two methods. Apart from the relevant Research question, this choice was also done according to the internal validation mentioned in Section 3.5 in regard to the possible abstraction problems [27]. To begin, the approach based on the n-grams method was a *substring* method. Similarly to the n-grams method, this evaluated data of length n, but instead of evaluating sentences of n number of words, it evaluated every sequence of n number of characters [9]. The implemented algorithm did, however, support any value n but given the common range of 3–6 as stated by Cohen, internal validation was done with three character substrings. As per the method description, counts were then done on the found substrings, which in this case was the same process as the basic word count, i.e., sorting the substrings alphabetically and counting them in one pass [9]. To decide the significance of words in each segment, the specific substring scores for each segment was evaluated first. In this case, each substring was considered the "sample" for scoring, while occurrences of the same substrings in the other segments (or documents if the entire corpus is considered) were seen as the "background". In addition to this, the total number of symbols in the segment, S, and the total number of substrings in the document, R, were evaluated to score each sample. The scoring function utilized in this implementation, was similar to the one presented by Cohen, but with the current sample removed from the background, an alternative mentioned by Cohen [9]. This approach was chosen since when the relevant segment (or document when the entire corpus was considered) was included in the background, the score would always become zero, for every substring. The implemented and used function used to score individual substring can be seen in Equation 4.1.

$$\psi_i = \begin{cases} C_i \cdot \ln(\frac{C_i}{S}) + B_i \cdot \ln(\frac{B_i}{R}) - (SC_i + RB_i) \cdot \ln(\frac{SC_i + RB_i}{S+R}), SC_i \geqslant RB_i \\ 0, SC_i < RB_i \end{cases} \tag{4.1}$$

, where C_i is the count of the current sample, B_i is the count of the same n-gram in the background, sans the sample count [9]. With this approach, it was thus possible to score the n-grams in relation to each segment, i.e., the sample is taken from a given segment, and the document is considered the background, or in relation to each document. For document-specific scores, the sample was instead taken from a whole document instead and the entire corpus was considered as the background. Based on the substring scores for each segment, Ψ, the words in said segment were then scored. Each word within a segment was split up

into every possible substring of the given length n, and the scores of present substrings in the word was summed to get a score for the given word, according to Equation 4.2.

$$\omega_i = \sum_{i=1}^{n} \begin{cases} \psi_i, g_i \in w_i \\ 0, \text{otherwise} \end{cases} \tag{4.2}$$

, where n is the number of n-grams for the segment, g_i is the current n-gram, and ω_i is the resulting score for the word w_i in the segment W_i.

The *TF-IDF* method, was, in the same manner as the n-grams method, implemented to support both segment specific and document-specific scores. Thus, the term frequency, *TF*, described, either, the frequency of a word in a given segment, or in a document as a whole, being calculated according to Equation 4.3 [11].

$$TF(w_i) = \frac{n_{w_i}}{n_W} \tag{4.3}$$

, where w referred to the relevant word, n_w referred to the number of occurrences of w within the segment or document W, and n_W referred to the total number of words in W. The *IDF* part of the method, on the other hand, is instrumental in lessening the significance on high-frequency words, while giving less frequent words a higher weight, for example, ensuring that conjunctions generally is scored lower than nouns. This was evaluated by dividing the total number of segments or documents by the number of segments or documents the relevant word was present in, according to Equation 4.4 [11]. These separate calculations were then multiplied to archive a final score for the relevant word w, as seen in Figure 4.5.

$$IDF(w_i) = \log(\frac{N}{N_{w_i}}) \tag{4.4}$$

$$\omega_i = TF(w_i) \cdot IDF(w_i) \tag{4.5}$$

, where N referred to the total number of segments or documents and N_{w_i} referred to the number of those that contained w.

As described throughout their descriptions, both these methods were implemented to also be able to score entire documents based on the corpus. This decreased "resolution", i.e., from singular segments within documents, to documents within corpora was done to be able to compare how similar two documents were to one another, based on present words and their corpus based scores. To do this, the cosine similarity method was utilized, where a vector of every word in the corpus was considered and scored for every document in the corpus. This meant that each document had an accompanying vector, where each number represented how significant the word w was to the given document. The positions in this scoring vector, all had the same index as the word it represented in the word list for the entire corpus, i.e., w_i in the list $\boldsymbol{W}$ would have the accompanying score τ_i in vector $\boldsymbol{T}$. Comparing two documents was then done by considering their scoring vectors as $\boldsymbol{A}$ and $\boldsymbol{B}$, where their similarity would be calculated according to Equation 4.6 [12].

$$\cos\alpha = \frac{\boldsymbol{A} \times \boldsymbol{B}}{|\boldsymbol{A}| \times |\boldsymbol{B}|} = \frac{\sum_{i=1}^{n} \boldsymbol{A}_i \times \boldsymbol{B}_i}{\sqrt{\sum_{i=1}^{n} (\boldsymbol{A}_i)^2} \times \sqrt{\sum_{i=1}^{n} (\boldsymbol{B}_i)^2}} \tag{4.6}$$

, where n was the length of the scoring vectors, i.e., the length of the list $\boldsymbol{W}$, and i was the current index. To have a method that cosine similarity could be compared to, *Jaccard* similarity was also introduced. Instead of utilizing the scoring vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, this used the corresponding word lists for the same documents, i.e., W_A and W_B. This approach aimed to simply divide the number of shared unique words between the documents to the total number of unique words. This meant that two identical documents would have a similarity of 1, while two completely different documents (documents containing different languages for example) would have a similarity of 0 [13].

$$J = \frac{|W_A \bigcap W_B|}{|W_A \bigcup W_B|} \tag{4.7}$$

, where J is the similarity between document A and B. Additionally, the similarity score was normalized based on the highest similarity to assign the connection with the highest similarity with the value 10. This was meant to allow the user to filter the connections with an easy-to-read slider from 0–10. A similar normalization approach was also done for the scores acquired either from Equation 4.2 or 4.5, according to Equation 4.8

$$\widetilde{\omega_i} = \frac{\omega_i}{\Omega_{max}} \cdot 10 \tag{4.8}$$

, where Ω_{max} referred to the highest value contained among the scores, Ω, in a given segment.

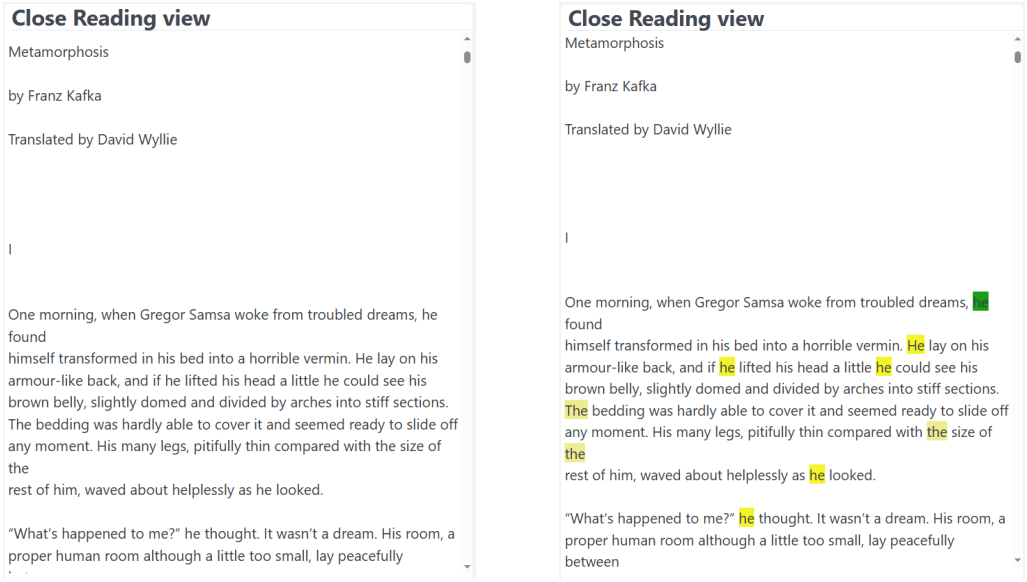
4.3 Close reading view

The first view in the visualization tool (when read from left to right), was the close reading view. This view utilized the plain text along with the extracted word array, mentioned in Section 4.2, which allowed the tool to both write out the text, and highlight words that would be accounted for in the text analysis. The highlighting method in this context was implemented as a simple keyword search method, i.e., when a word is either hovered on or clicked, all occurrences of said word would be found in the document and highlighted. A similar approach to the highlighting functions featured in *Voyant tools*, *VarifocalReader*, and *uVSAT* [3, 45, 46]. This highlighting method was also connected to the frequency, and trends view with the aim of facilitating the *locate*, *distinguish*, and *compare* tasks mentioned in Section 3.2, all from hovering on a specific word. In addition to this, to provide the user with the ability to interact with other views while a word was highlighted, clicking on a word "marked" it, i.e., saving the highlight for the user to stop hovering on it, and still having the same information available. Both the "plain" view, without words highlighted, and a view where the word "He" has been marked and the word "The" hovered can be seen in Figure 4.1. In addition to the highlighting feature, the close reading view was also implemented to include segment breaks to show where the segments in the trends view begin, an example of which can be seen in Figure 4.2. These segment breaks were also colour-coded similarly to the segments in the trends view to make it clearer what they represented.

As mentioned, the close reading view was generated from both the plain text of the input file, and the raw data array. This approach was chosen, given that it allowed for line breaks and other non-letter symbols to be kept in the text, as seen in Figures 4.1–4.2, along with only highlighting the present words from the raw data, as seen in Figure 4.1b. Thus, nothing would be highlighted if the user, for example, hovered over any form of punctuation or number. The segment breaks also utilized the raw data array by counting the number of words passed and only inserting a segment break once the correct number of words had been passed, according to Equation 4.9.

$$N_{Sw} = \lceil \frac{N_{Cw}}{N_S} \rceil \tag{4.9}$$

, where the result, as indicated, would be rounded up to avoid decimal values. Here, N_{Sw} referred to the number of words each segment would contain, N_{Dw} referred to the number of words in the document, and N_s referred to the number of desired segments. Apart from the interaction tasks that were facilitated for, these two highlighting methods were also included to take **RQ1** into account, aiming to figure out if they were suitable methods to present the location of specific information within a document.



(a) The close reading view without any highlights. (b) The close reading view with both marked and hovered words.

Figure 4.1: Example of functionalities in the close reading view.

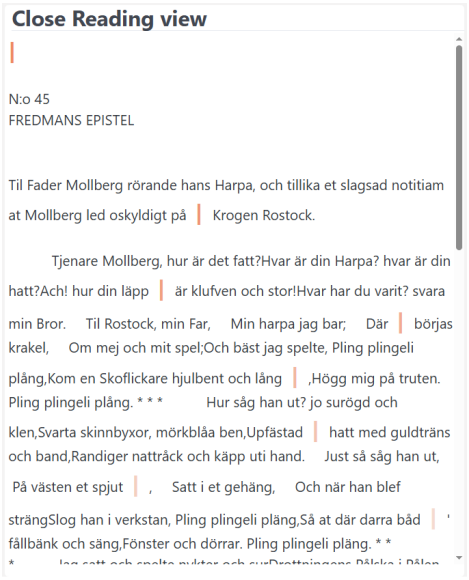
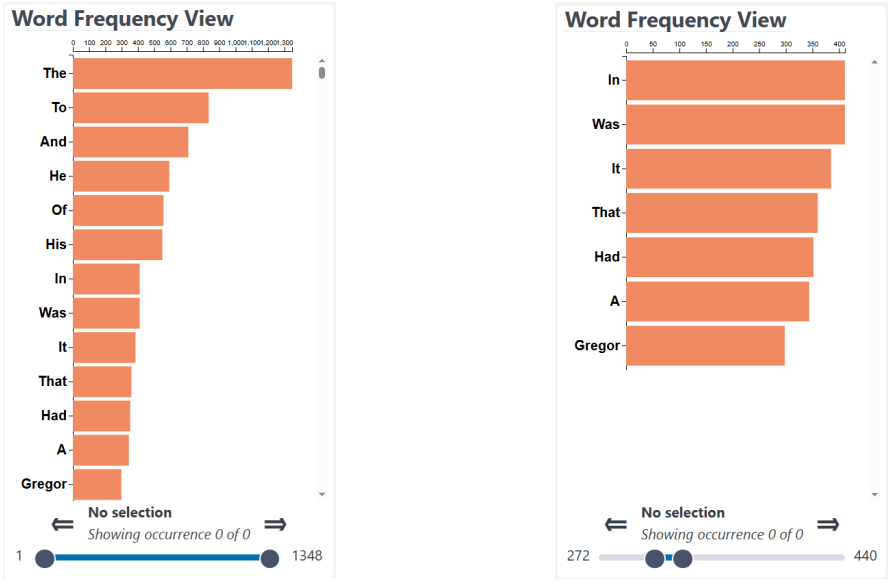


Figure 4.2: Example of the close reading view with very short segments.

4.4 Word frequencies

As described in the second paragraph of Section 4.2, the raw data array was fed into the basic word count algorithm to get all unique words in a text and their counts. This data was then used to generate a vertical bar chart with the bar lengths corresponding to the count of a given word. This also was done as part of Objective **O4** in Section 1.5, and mainly utilized *D3.js* to draw the graph, according to the description of the utilized tools in Section 1.6. The bar chart was generated utilizing the *D3.js* functions `scaleLinear` for the x-axis and `scaleBand` for the y-axis, where the domains were set from 0–*max word count* and 0–*the number of words* respectively [47]. Since it was figured that less frequent would be harder for the user to find, a dual range slider was also introduced to either filter out, for example, the most frequent



(a) The plain frequency view.

(b) The frequency view after occurrence filtering.

Figure 4.3: Example of functionalities unique to the frequency view.

words or words with few occurrences. An example of the plain view along with the same view, but filtered, is shown in Figure 4.3

In addition to this, the frequency view also featured a similar highlighting feature as the one mentioned in Section 4.3, but with the addition of a tooltip to show the hovered word and the exact number of occurrences of said word. Still in accordance with **RQ1**, as with the highlighting featured in the close reading view, this was connected to said view through its own highlighting functionality. What this aimed to provide to the user was a way to rank the words present in the text, along with making it possible to quickly locate them in the close reading view, two relevant user objectives as described in Section 3.2. For words that had few occurrences, and especially if they were far apart in a given text, two arrows were also implemented to let the user quickly scroll to the previous or next occurrence with single clicks. An example of the connection between the two views can be seen in Figure 4.4

4.5 Word trends

To generate the word trends view, different data compared to the close reading and frequency view, was required. As described in Section 4.2, words contained within a segment of a given document was ranked on their significance within said segment. As previously mentioned, these significance scores were also normalized according to Equation 4.8, ensuring that the most significant word in a given segment would always have the score 10, for all scoring methods. This data was then used to generate a stacked horizontal bar chart where the x-axis corresponded to the different segments and the y-axis corresponded to the cumulative score of the top words in said segment. It is important to note here that "top words" referred to a value the user could decide themselves, which by default was set to 10, thus showing the ten highest scored words in each segment. This view was also implemented using the `scaleLinear` and *scaleBand*, as with the bar chart described in Section 4.4, but with the x-axis corresponding to the band scale, and the y-axis corresponding to the linear scale to make it horizontal [47]. Here the x-axis was divided in an ordinal order, where each position corresponded to a segment in the document, divided with the same approach as described by Equation 4.9. This ensured that the segment breaks mentioned in Section 4.3 would correspond to the different positions on the x-axis in the word trends view. The choice to utilize a

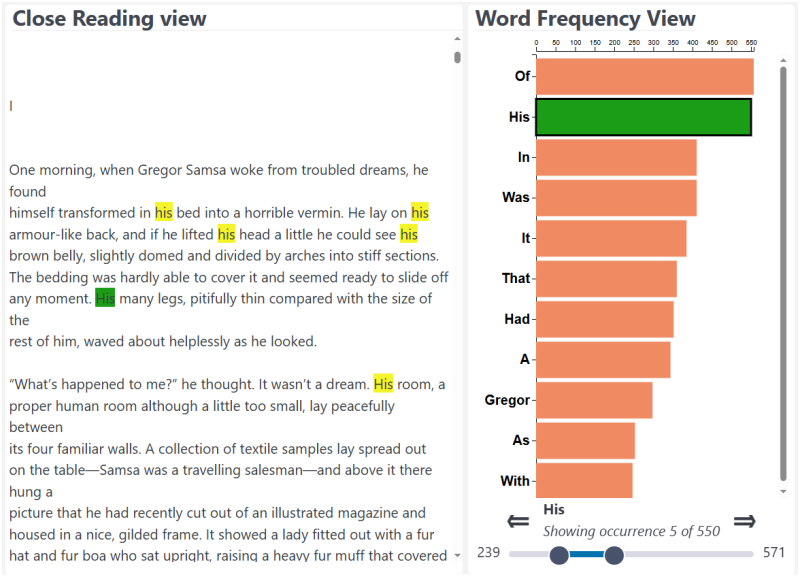


Figure 4.4: Example of the connection between the close reading and frequency view once a word has been selected.

stacked bar chart for the view was based on the discussions and concern in Section 3.3 about the available display size. It would be possible to assign each word in a segment to a unique position on the x-axis, but depending on the number of chosen segments and top words, the user would have to scroll horizontally to see if the same word is significant in a later segment. This means that the possibility of tasks related to comparison and correlation would be decreased given the loss of information from the user scrolling away from relevant information. As mentioned, it was therefore decided to utilize a stacked bar chart instead, to reduce the required size of the graph while retaining the same information. Since the words significant to a given segment were stacked on top of one another, the total height of each bar corresponded to the cumulative score of the n top words in said segment. Thus providing the user with information about which segment had the highest concentration of significant words. This view was additionally implemented with Shneiderman's *Visual Information-Seeking Mantra* in mind, i.e., to firstly provide an overview, then allow the user to zoom and filter, and finally provide details on demand [30]. For the overview part, colours were utilized, separating the segments into a gradient from orange to white to blue, a palette acquired from *Colorbrewer* [48]. The colouring of each separate word in the graph was dependent on which segment it first was significant for, i.e., if a word was significant to the first segment, it would get an orange colour while a word that was only deemed significant to the last segment would be assigned a blue colour. This approach was chosen with **RQ1** in mind, providing the user with a quick overview of where certain words could be found in the document. Both the plain view generated with 45 segments and 10 words accounted for, along with the colour scheme for this, can be seen in Figure 4.5.

As indicated by the mantra, the user also had the possibility to "zoom and filter", which in this case related to the number of top words and the number of segments the user chose to include. These were available through the settings sidebar of the tool, where the different scoring methods also were selectable. For example, if the number of segments was decreased, there would be a clearer distinction between the segment bars, thus zooming out or decreasing the resolution of the presented data. Changes to the number of top words presented were instead meant to serve as a way for the user to filter the number of displayed words, since the size of each word in the stack was scaled according to the number of words within it. An

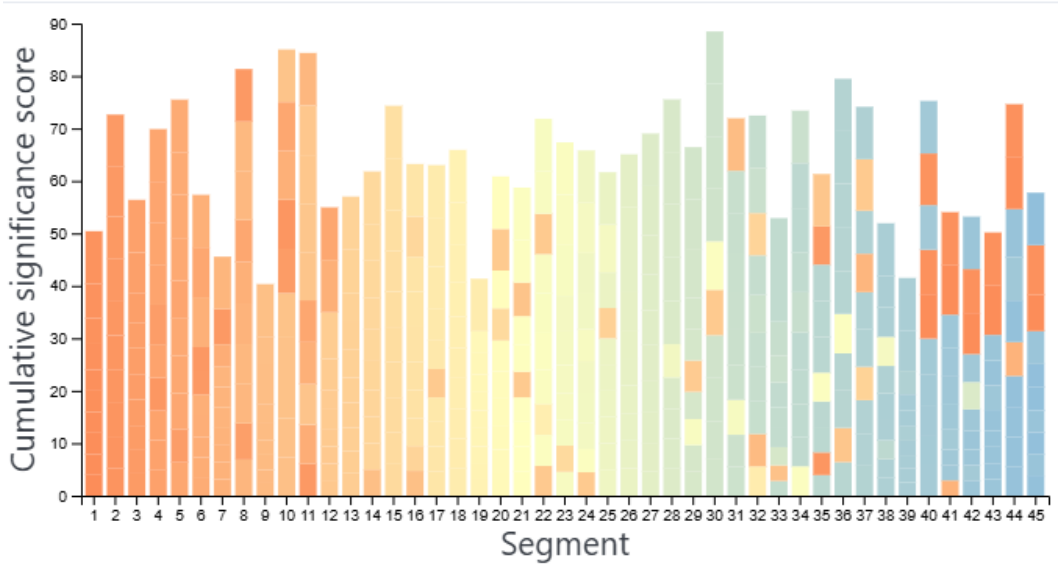


Figure 4.5: Example of the word trends view, generated with the default settings.

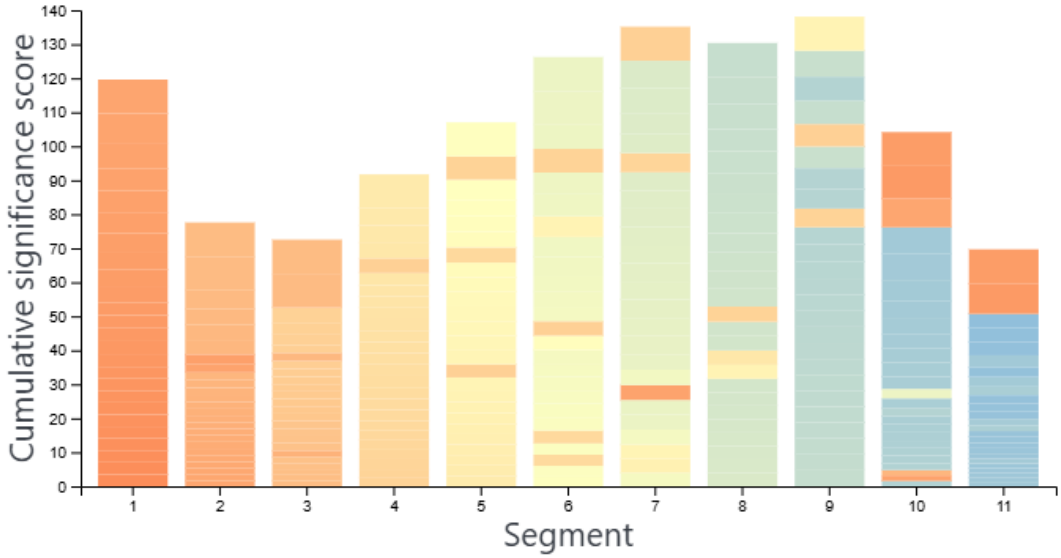


Figure 4.6: Example of the word trends view after "zoom and filtering".

example of the view, when the number of segments has been decreased to 11 and where each segment bar presents the 25 most significant words to said segment, can be seen in Figure 4.6 Details about the words contained in the view were provided through a similar highlighting approach as the ones presented in Sections 4.3 and 4.4, while also providing what score the word had within a given segment. As with the previously mentioned views, this also relied on the user hovering over a specific entry in the graph to highlight it, and if they clicked, the highlight would be saved. An example of the highlighting within the view, along with how it affects the other two "single document" views, can be seen in Figure 4.7, where the highlight for the word "Bed" has been saved, and the word "His" is currently highlighted through hovering.

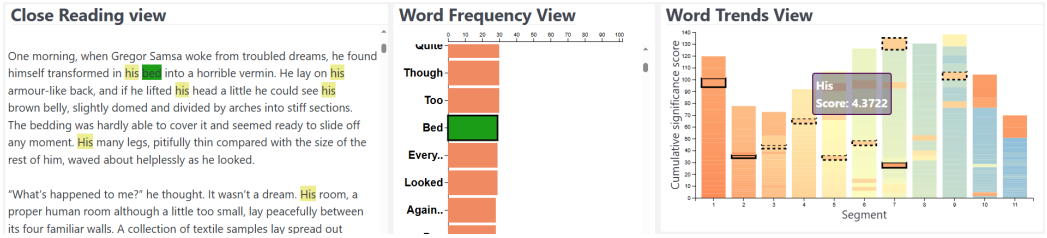


Figure 4.7: Example of highlights across all single document views.

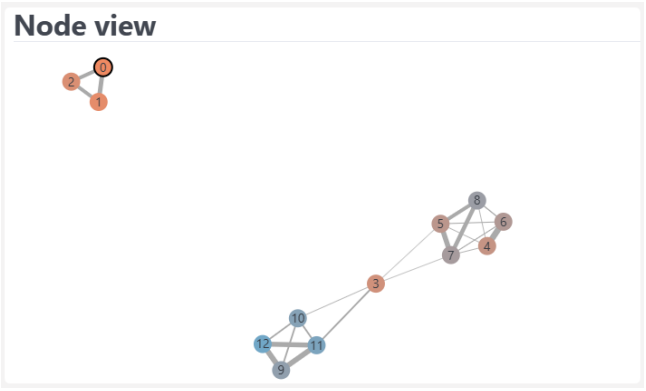


Figure 4.8: Example of the node view.

4.6 Corpus node graph

To allow for objectives like clustering and distinction, as presented in Section 3.2, a node graph containing all documents of the corpus was also implemented. This did, however, differ from the initial ideas for such a graph discussed in Section 3.4, where instead of connecting documents, or nodes in this case, based on shared word significance, it was chosen to evaluate the similarity between all document pairs with methods described in Section 4.2. The design in this view was also based on the visual information-seeking mantra, where the overview relied on colouring the different nodes in unique colours and highlighting the currently viewed document to allow for distinction. The overview idea was also strengthened by writing the document indices, based on the load order to the corpus, in the centre of each node. In addition to this, the links between nodes were scaled according to the similarity between those so that the user could get some idea of which documents were more similar to one another. Since this graph was presented on a two-dimensional plane, without distinct axes, the user could zoom and pan the graph to focus on desired nodes while a filtering method was accessible from the settings menu. This allowed the user to set a similarity threshold, filtering out connections that they deemed to be irrelevant to them. An example of a node graph created from this method can be seen in Figure 4.8, where the selected document has index 0 and a black border. Here it can be seen that two clusters have been formed, where the smaller group represented Edgar Allan Poe's works: *The Cask of Amontillado*, *The Fall of the House of Usher*, and *The Raven*. The other cluster on the other hand, represented several economic reports on different subjects from the Office for National Statistics, both collections being part of the corpus mentioned in Section 4.1.

4.7 Similarity overview

An additional feature implemented that had not been covered by the initial design plan described throughout Chapter 3 was the similarity overview. This was implemented to provide the user with additional information that highlighted why two documents were deemed to

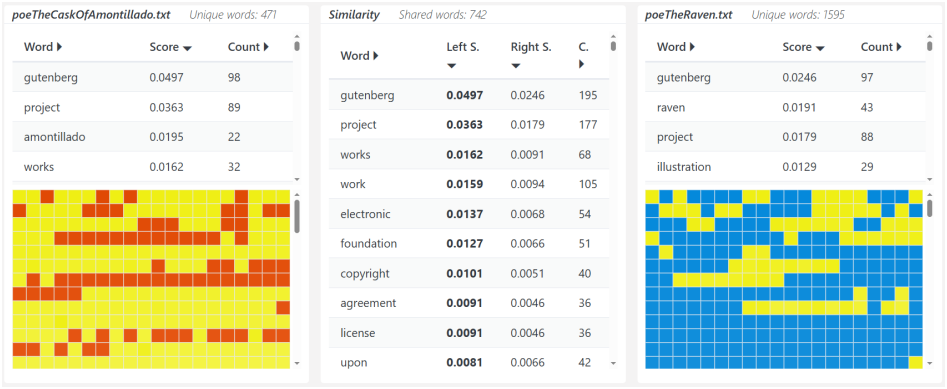


Figure 4.9: Example of the similarity overview.

be similar. This was made accessible through click interaction with any given edge presented in the node view seen in Figure 4.8 and opened a separate view to display the additional information available. The overview presented consisted of three columns; the left- and right-hand sides represented the relevant documents, and the central column represented shared words between these two documents. Most of the information was decided to be presented in a "plain" manner with tables that showed all relevant words, their corpus-based scores (as mentioned in Section 4.2), and their number of occurrences. To provide the user with a "quicker to read" overview, however, the left and right columns also featured grid-based representations of the words contained within the relevant document. These grids had the aim to quickly show the user the number of words that were unique to one document along with the number of words that were shared, and to further highlight the significance of the words within the document, the grids were sorted in a decreasing order based on the corpus-based scores. An example of two documents being compared can be seen in Figure 4.9 where unique words for the left-hand side document are displayed in orange and the unique words for the right-hand side document are displayed in blue. Shared words are displayed in yellow, based on one of the three value diverging schemes found on *ColorBrewer*, to ensure it stood out from both the orange and blue, along with the white background [48].
In Figure 4.9 it is also possible to see the arrows featured in the column headers of the tables. These were included both to show the user how the tables were sorted at any given moment and to allow the user to sort the tables based on any column through clicking on any given header. It was also chosen to display the larger score for each word in the central column to make that information more distinct to the user.

4.8 Corpus overview

The views described in Sections 4.3, 4.4, and 4.5 all aimed to provide different *document-specific* overviews, but the tool only provided *corpus-specific* information through the views described in Sections 4.6 and 4.7. To also provide a general corpus specific overview, a separate view with that purpose was implemented, similarly to the approach found in the *Text Visualization Browser* [49]. This view thus aimed to collect every document in the corpus and display them as separate cards that contained information about the relevant document and gave the user a way to navigate to desired documents by clicking on any card. Each card also had the same index as the nodes described in Section 4.6 written before the document name to provide the user with a written connection between the two views. An example of how the corpus overview was presented can be seen in Figure 4.10, where several documents have been loaded, each with a *fingerprint*, drawn with the same approach as described in Section 4.5. Utilization of fingerprints was chosen as a visual identification for each document, based on the ideas presented by Keim and Oelke on literature fingerprinting [50]. Compared

Figure 4.10: Example of the similarity overview.

to the dense information found in their approach, however, this was meant to provide the user with an abstracted structure of a document, mainly if certain significant words were repeated across the text, or if certain segments were more significant than others. Additionally, given the smaller size of the bar chart and that the user was not expected to interact with it, compared to the word trends view described in Section 4.5, the axes and their labels were removed.

4.9 Stop word inclusion

Given the aim to provide generalised text analysis to the user, as described in Section 1.3, it was initially meant to avoid the usual approach of filtering out stop words [6, 14, 11]. In this case, given that there was a risk that the analysis methods would become language specific and that the approach could be domain-dependent [9]. However, through internal validation of the methods described at the end of Section 4.2, significance was still given to prepositions; both for texts in English and Swedish, it was chosen to implement a way to add stop words. The previously mentioned aim of the tool was still kept in mind, and based on that, it was chosen to provide the user with a customisable stop word list to include when the corpus was processed. Through a separate dialogue, the user could freely enter words that they did not want to be part of the analysis in a text box, where the input would be filtered according to the *regex* approach mentioned in Section 4.2, which ensured that the user could separate the words as desired (spaces, commas, or periods, for example). What this method also allowed for, given that it made no assumptions on *what* the user entered, was multi-language support, since every word entered would be applied to the corpus as a whole.

Expanding on this idea, it was also chosen to provide the user with a set of predefined stop word lists covering several different languages. These predefined lists were acquired from Fergie and Eklem's library *stopword*, hosted on *npmjs*. This included 62 different language lists, whereas 61 were included in this tool since one of the lists was for numerals, which already was covered, as described in Section 4.2. What this library also included was a function to remove stop words based on a given array, which was used to filter the word lists of the documents in the corpus [51]. The different lists in the library were labelled with three-character-long language codes according to the *ISO* 639 standard; thus, to make it easier for the user to find desired languages, it was chosen to translate the codes [52]. Conversion between the codes and the name of the relevant language in English was implemented utilizing the *iso-639-3* library written by Titus Wormer, also hosted on *npmjs* [53]. The combination of predefined stop word lists and free-form input from the user was implemented by "loading"

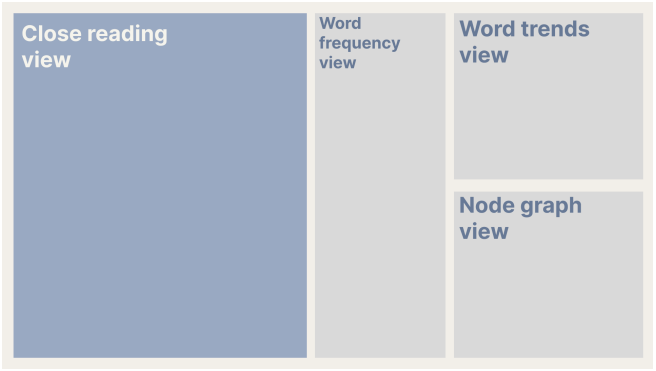


Figure 4.11: Initial layout mock-up.

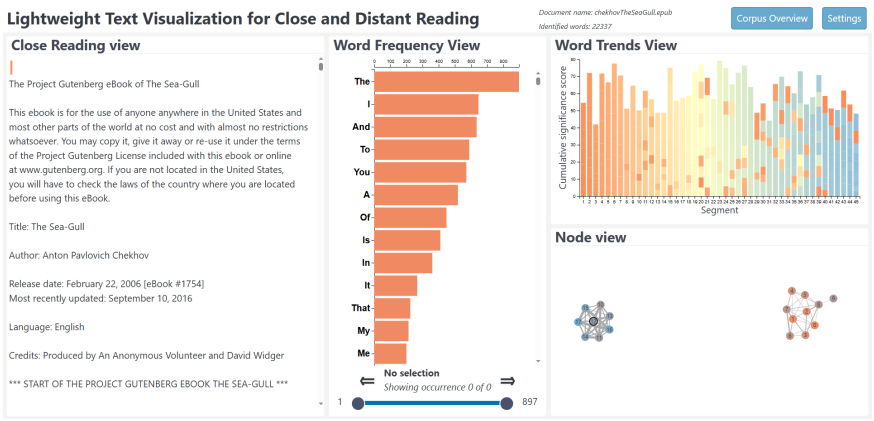


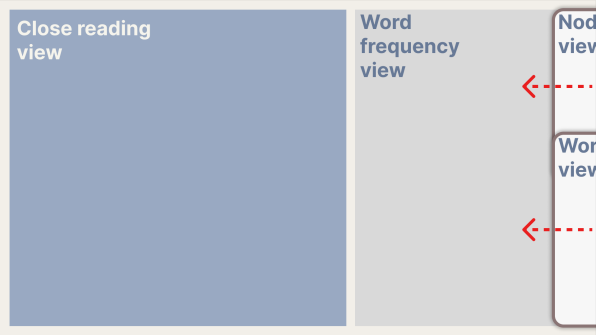
Figure 4.12: The implemented layout.

the desired list into the text box for user input, which also meant that the user could remove undesired stop words from the predefined lists. Lastly, it should be noted that this method was implemented to allow the user to freely disable stop word filtering by clearing the text box, which also was the default setting for the tool.

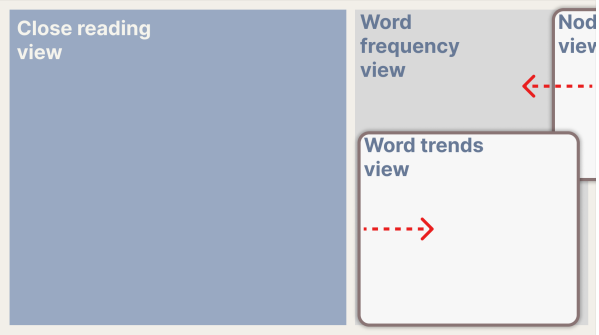
4.10 Layout of the visualization

The interface for the tool was implemented to combine the close reading view described in Section 4.3, word frequency view described in Section 4.4, word trends view described in Section 4.5, and node graph view described in Section 4.6 into a "main view" for the tool. For simplicity, this will be referred to as the "layout" of the tool onwards, which defines which views went where within the available display space. Given the motivation defined in Section 1.2 the tool was meant to provide the user with both close and distant reading methods, and based on that, an initial layout was designed. This layout can be seen in Figure 4.11, which also served as the mock-up for the design that was implemented. The general idea behind this layout, i.e., to have the close reading view and several distant reading views visible at the same time, was based on approaches found in other text visualization tools such as *DosVis*, *Voyant Tools*, and *Lariat* [3, 4, 7]. In the implemented layout, the influence of the mock-up was clear from the beginning, since it, apart from the colour palette and addition of a header, was implemented with the same idea, as can be seen in Figure 4.12. It is also important to note that the aspect ratio is different between the two layouts, since the available display space was affected by the toolbars in both *Google Chrome* and *Windows 10*.

This initial layout was implemented with **RQ3** from Section 1.4 considered, which was planned to be answered through the implementation of alternative layout examples as well.



(a) Additional views closed.



(b) Additional view opened.

Figure 4.13: The mock-up for the first alternative layout of the tool.

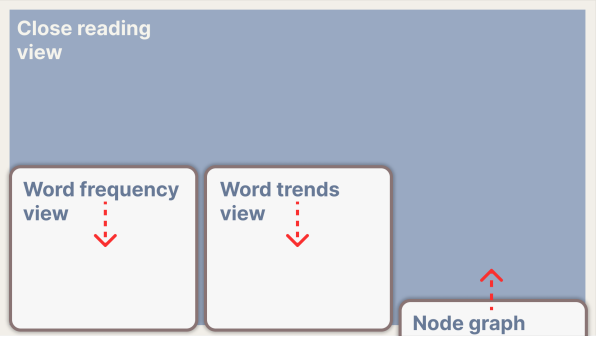
In this case, the mock-up seen in Figure 4.11 was considered a *full overview*, i.e., it always displayed all views mentioned in this subsection to the user. The other layout approaches would be to provide certain views to the user on demand, either with two views or only one constantly visible. What these alternative layouts aimed to accomplish was to provide the user with larger views compared to the initial layout, while still accounting for the limited display space. Thus, instead of having static views, the user was meant to be able to slide desired views into the display space, either vertically or horizontally. The first alternative approach kept the close reading and word frequency views as main views, while the word trends and node view would slide in from the right-hand side on demand. The mock-up for this approach, drawn similarly to Figure 4.11, can be seen in Figure 4.13, where the red arrows were meant to show the direction of the movement. The second alternative approach on the other hand only kept the close reading view as the singular main view, while the other views could be opened on demand. This approach can be seen in Figure 4.14, where it can be seen that, compared to the first alternative layout, the distant reading views would slide in horizontally from the bottom.

One thing accounted for in both of the alternative views was the principle of psychological closure. [54] Thus, they were intentionally positioned outside the available display space to tell the user that they could be opened, without explicitly writing it out. Additionally, as with the implemented version of the initial layout seen in Figure 4.12, the actual resolution would be different, along with the addition of a header.

4.11 General interaction techniques

The interaction techniques covered in Sections 4.3, 4.4, 4.5, 4.6, and 4.7 were generally *view-specific* and implemented in different ways. Other interaction techniques that effected multiple views, or the tool as a whole, was considered *general interaction techniques* instead, and will be referred to as such henceforth. Firstly, for views whose contents exceeded the avail-

(a) All distant reading views closed.



(b) Two distant reading views opened.

Figure 4.14: The mock-up for the second alternative layout of the tool.

able height of the element, scroll bars were made available to the user, for example found in the close reading and word frequency view along with the corpus overview. This enured that all available information was provided to the user, while entries and text still was displayed in a readable size. Horizontal scrolling was, however, decided not to be implemented, even though it could be applicable to allow for wider bars in the chart. The reasoning behind this was based on the idea that significances for words that appeared in the beginning and end of a given document should be comparable at the same time. Thus, the bar chart was instead implemented to fit the view, and to provide the user with all information simultaneously. Moving on, the navigation between different documents in the corpus was also implemented as a general interaction technique, since the three available methods to do so, relied on the user clicking on the desired document to load it. To also signify that this technique was available to the user, all places were it could be used was indicated by the index number, two examples of this mentioned in Sections 4.6 and 4.8. Apart from those views, this technique was also made available from a drop-down menu in the settings bar, also including the indices, but with the requirement that the user pressed the "confirm" button to change document.

On the subject of the settings bar, this included several general interaction techniques, used to change how certain things in the views were displayed, such as the sliders for number of top words and segments mentioned in Section 4.5. Similar sliders were also implemented for other techniques that provided the user with a way to change numerical values in the tool, mostly values used for filtering, but also values tied to the text analysis methods. In addition, to change between analysis methods the user was provided with drop-down menus which contained options for the methods described in Section 4.2. To note about the reason why drop-down menus were implemented for method selection was since it provided a simple approach for further implementation of other analysis methods and inclusion of them in the settings bar. Moving back to the confirm button in this area of the tool, it was implemented to help with performance while the user used the tool. In this case, the idea was that the user

would change all settings they wanted, confirm the changes, and then use the tool with less re-analysis of the corpus.

Lastly, the file or text input approaches were also considered as general interaction techniques, since they were made available from two different parts of the tool. For file input, the technique utilized the standard input method for files that was provided with *HTML*, specified as a multi-file input and restricted to only allow file formats that were supported in the tool. This ensured that the user quickly could load an entire corpus, as well to avoid possible issues if unsupported file formats were loaded.

5 Evaluation

As highlighted in Section 3.6, the thesis work covered evaluation of the implemented tool described in Chapter 4. However, the user free evaluation, mentioned in Section 3.6, was still part of this broader sense of evaluation, and aimed to handle internal validation of the tool. The user tests, aimed at evaluating the tool as a whole, were conducted with participants that fit the target user base as described in Section 3.1, while the limitations described in Section 1.6 were considered as well. This approach aimed to ensure that suitable data was collected, where the decision whether a test contained suitable data or not was answered with the help of a *Mini-VLAT* assessment test each participant completed [55]. Along with the *Mini-VLAT* assessment, the test participants were also asked about their age, educational background, and language proficiency, both to take the points relevant to evaluation in Section 1.3 into account along with ensuring that the participant could understand the written text in the tool. The collection of this demographic data was also done with the target user base from Section 3.1 in mind. It is important to note, however, that the visual representations and interaction techniques were tested during the same tests to save time, but their results were evaluated separately.

5.1 Validation during the implementation stage

During the implementation process, there was a need to validate that certain methods produced the correct results, to avoid problems that related to faulty algorithm design, as discussed in Section 3.5 [27]. This validation step also aimed to ensure that the user tests would be conducted with a working tool, since results from buggy implementation would be difficult to compare with more refined implementations [56]. This internal validation referred the process where it firstly was determined if the *NLP* methods described in Section 4.2 worked as expected and the time it took to handle the documents and entire corpus with the different methods. For this, the extraction of words worked if it correctly separated all words from a text while validation of the analysis methods related more to how suitable the scores of words were based on the segments, or how well similarity was decided. This was a more simple approach to validation compared to the definition used by Riezler and Hagmann, since their approach related to machine learning, but some ideas, as described were still utilized here [15]. Some of these validation points were quantitative in nature, for example number of words correctly extracted, or the correct counting of word frequencies. Other points was

however qualitative since they related to if the analysis felt correct or suitable based on the corpus and its contents. These points of validation were included with **RQ2** in mind, to decide if one method was better suited for the tool or if it was more "lightweight", based on the described motivation and aim of the tool presented in Sections 1.2 and 1.3. What all points of validation aimed to achieve though was to ensure that the methods implemented worked to the degree were the views described in Chapter 4 could be produced according to the aim described in Section 1.3.

5.2 Use case

Based on the described processes throughout Chapter 4, two use case diagrams were defined. The images in these diagrams are taken from the second prototype of the tool, and the second diagram does also include functionalities added to this version. To give some sort of distinction between the two prototypes, however, the following subsections will mention if a specific functionality was implemented for the second prototype. Thus, if the second prototype is not mentioned for a specific functionality, it is present in both the first and second prototypes.

5.2.1 Use case on tool startup

The first one, seen in Figure 5.1, aimed to describe how the tool handled the initial corpus provided by the user through the welcome dialogue shown on startup. As seen in the darker orange *Initial corpus input* area, this mostly came down to the choice of whether the user would use the default settings or if they wished to change certain parameters. To note, however, is that both options lead to the same "*NLP* pipeline", which began with the lighter orange *Document preprocessing* area. In that part of the process, the submitted documents were first converted to plain text and then had their words extracted, according to the methods described in Section 4.1 and the beginning of Section 4.2 respectively. As can be seen in the diagram, the results of these processes were also used as data for the close reading view. The plain text used to generate the written text while the extracted words were used to decide which words would be possible to highlight. After the preprocessing, each document was then evaluated with the parameters chosen by the user. This can be seen in the yellow area labelled *NLP analysis* and done according to the methods described in the second part of Section 4.2. As seen in the diagram, the word count from each full document was used as data for the word frequency view, while the result of the document segmentation was used to place segment breaks in the close reading view. The word scores within each segment were, on the other hand, used as data for both the fingerprints found in the corpus overview and the word trends view. Finally, the corpus-based scores from the cross-corpus occurrence counts, were used as data for the similarity overview and node graph view. With the data collected in the green area labelled *Data mapping*, the interface of the tool was drawn and presented to the user, where the initial document displayed always was the one with index 0 in the corpus.

5.2.2 Continuous use case

As mentioned, this described the startup process of the tool; once loaded, the user was meant to use it continuously, according to Figure 5.2. This diagram is centred on interaction with the interface, with the *NLP pipeline* slimmed down since it followed the same structure as seen in Figure 5.1. Compared to the startup process, smaller red arrows were featured here to show views or windows that could be opened and closed on demand. Beginning on the top row, on the right-hand side of the *User interface* area, was the settings bar. Firstly, this had the option for the user to provide additional documents to the corpus through a dialogue window, which would send the updated corpus through the *NLP pipeline*. Similarly, the inclusion of stop words was also done through a separate dialogue window, according to the

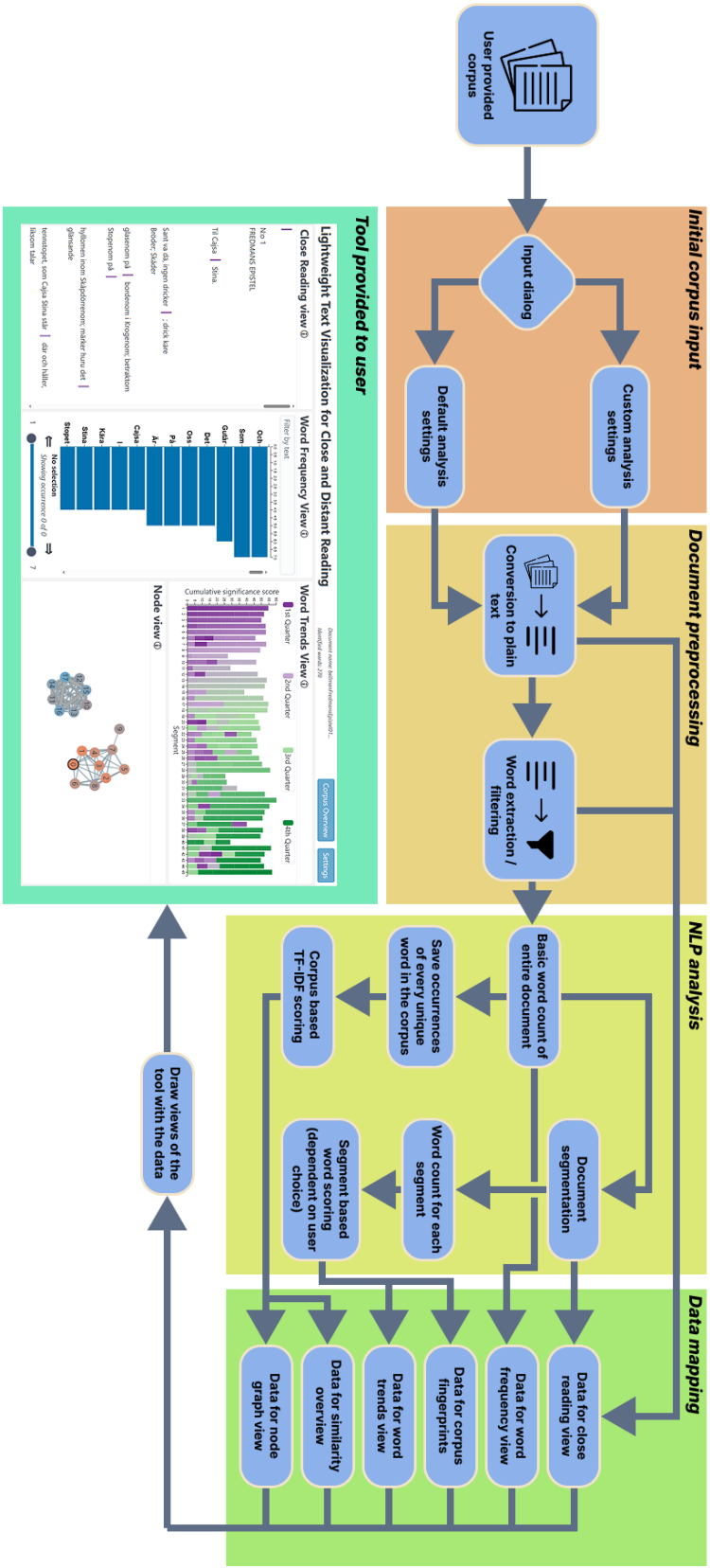


Figure 5.1: The use case diagram from the point of the user submitting their corpus up to initial view generation.

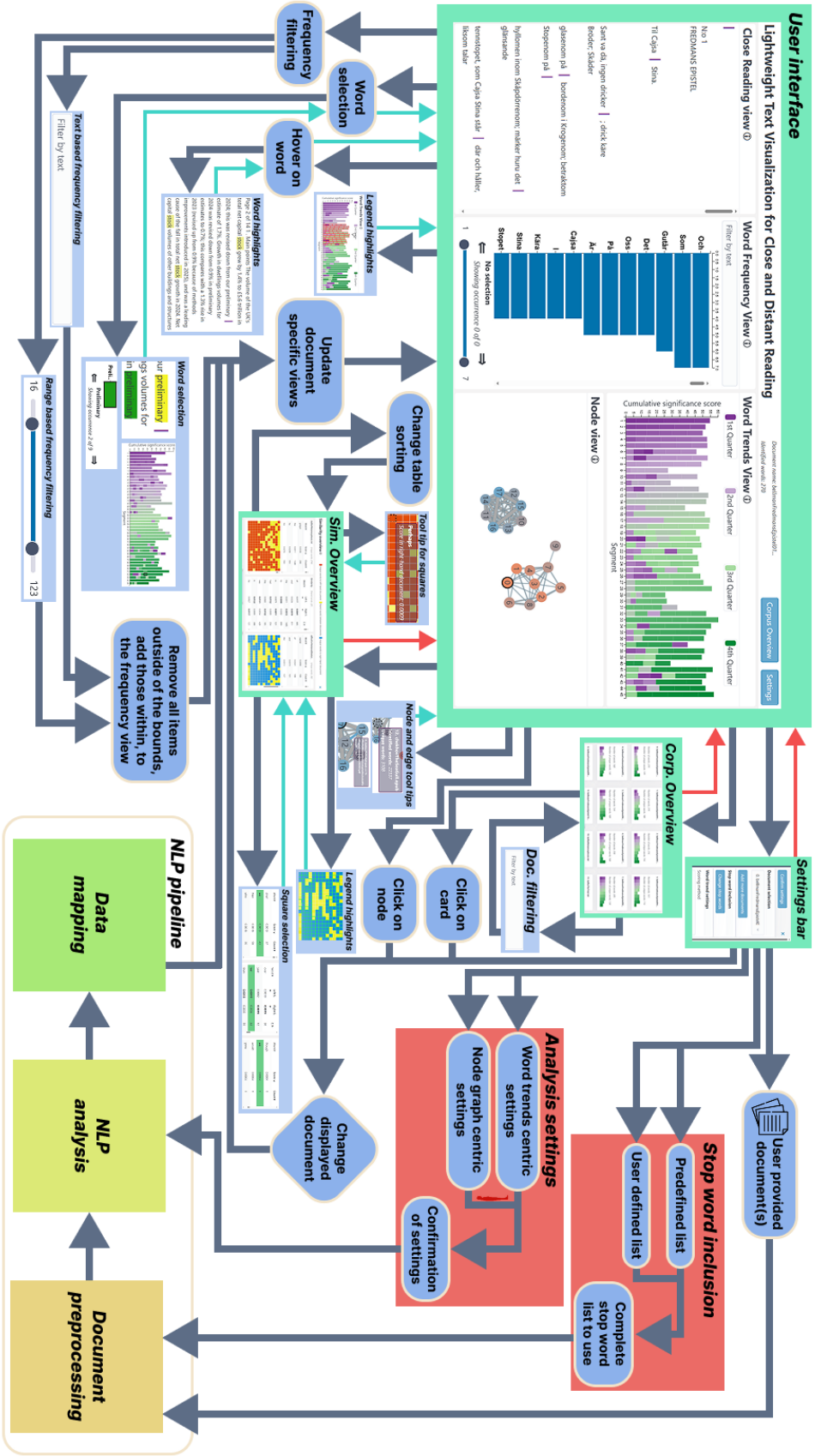


Figure 5.2: The use case diagram for continuous use of the tool after the process in Figure 5.1 has finished.

method described in Section 4.9. On confirmation, regardless of choice, the corpus would be sent through the *NLP pipeline*, with the added stop words as filter parameters. As can be seen in the diagram, either the addition of new documents or stop words, would send the new data at the beginning of the *NLP pipeline* since these changes required preprocessing of the documents. Apart from the inclusion of new data, the analysis settings were also present in the bar, meant to provide the user with choices on how the documents and corpus were analysed. On confirmation, the new parameters were instead inserted in the middle of the *NLP pipeline* since no preprocessing was needed to update the present data. Lastly, as mentioned in Section 4.11, corpus navigation was also possible from the drop-down menu in the settings bar. If used, no new data had to be analysed and thus the document-specific views were simply updated according to the data already present for the selected document.

Next to the button for the setting bar was the button for the corpus overview. As mentioned in Section 4.8, this dialogue window featured slimmed down information about each document in the corpus as separate cards. If the user clicked on any of the cards, the document-specific views would be updated accordingly, without any additional analysis of the corpus or document. An additional feature of the corpus overview, added with the second prototype, was the text-based document filtering functionality. This would take the provided pattern from the user and hide cards that did not contain it from the view. Similarly, if the user clicked on any of the nodes in the node view, the same process would execute for the relevant document. To get additional information about the nodes and edges in this view, users could hover on either to display the relevant document if a node was hovered or the strength of the relevant pair if an edge was hovered. With this functionality, teal arrows are also included in the diagram, signifying the showing and hiding of highlights or selections, which stays relevant for the rest of the diagram. Also, accessible from the node view, as described in Section 4.7, was the similarity overview, which featured information about the connected documents. This view did feature some interactive elements through hovering, providing a tooltip with information about the hovered square. The most notable feature, however, was the selectable ways to sort the tables. If done, this would redraw the tables based on the desired column and order to sort by. In the second prototype, though, additional functionalities were also implemented, providing legend highlights, quickly showing which words were unique to each document or which words were shared between them. Additionally, the squares were also made clickable in the second prototype, which would scroll the tables the clicked word was present in, thus providing the user with a quick way to find desired words.

Accessible from the user interface and present across most of the main views in the tool were interaction techniques related to the document-specific views. As seen in the diagram, the available techniques were frequency filtering, word selection, word hovering, and implemented in the second prototype, legend highlights to the word trends view. Starting from right to left, the colour legend of the word trends view would, similarly to the legend highlights in the similarity overview, highlight the segments that are part of one of the quarters of a given document, quickly giving users a way to see which segments each colour represented. If the user hovered on any word in the close reading view, or any bar that represented said word in the frequency view or trends view, would highlight the word across those views. Similarly, if the user clicked on any word or bar, this word would be selected, both shown through colour and added outlines, with the selection also being written out in the frequency view. Lastly, the frequency filtering was the only technique of these three that would change the displayed data in the document-specific views. Here, based on the user-selected upper and lower bounds of the dual range slider, the frequency view would be updated to only show words with occurrences within the selected range. This was also reflected in the close reading and word trends views, since omitted words were no longer able to be highlighted or selected in those views. In the second prototype, however, another method to change what data was displayed in the document-specific view was also implemented. This was a text based filtering method that would work with the range-based slider to only display words within the defined range and that contained the user-provided pattern.

5.3 User test interview

The user tests were planned as an interview, where the participant was tasked with completing different tasks in the tool and answering questions on how they believed certain things worked within the tool. What the following descriptions aim to describe is the exact procedure that was used when conducting user tests. The entire plan, as it was written, can be found in Appendix A. Firstly, the test participant was introduced to the thesis through a simple retelling of the general motivation and aim behind the work, based on Sections 1.2 and 1.3, and after that was asked if they would like to partake in the test, even though they had agreed on it beforehand, seen in A.1. This aimed to somewhat avoid problems that related to the participants' familiarity with *traditional interfaces*, helped through a description of what the tool can be used for [56]. In addition to this, the short introduction was meant to provide the test participant with most of the relevant information before the test started, a choice based on Carpendale's advice for the interviewer to talk less [20]. After the introduction, and the test participant's agreement to partake in the test, they were asked about their age, educational background, and English proficiency, as mentioned in the introduction of Chapter 5, seen in A.2. The final step before the participant was shown the tool tasked them with completion of the *Mini-VLAT* assessment test, where it was explained that it would take approximately five minutes, seen in A.3. This assessment was included as a way to measure how familiar the test participant was with general visualization methods since this could skew the test results [55].

After completion of the assessment test, along with the preceding tasks, they were provided with the tool, which was set to display the main view (as shown in Figure 4.12), seen in A.4. Important to note here is that the corpus used for each test was identical and always loaded in advance, which ensured all participants had access to the same data and would not have to wait for the load time. The titles, authors, sources, and extracted word counts of the documents within this corpus can be found in Appendix B. The first task related to the tool was to task the user to freely navigate within the tool, meant to provide them with familiarity with the general layout of it, seen in A.5. However, the task was also included to evaluate how distinct each view was from one another, along with how easy it was to find the two views "hidden" behind clickable elements, i.e., the corpus and similarity overviews. The hypothesis here, based on Carpendale's *Quantitative Methodology*, was that the implemented layout and the titles for each view would be enough information to tell the user that different data was displayed in said views [20]. Based on the limited interaction the participant would have got at this point, they are then simply tasked to provide an explanation on how they believe the different views are connected. According to A.6, this aimed to study if it was possible for the participant to *understand* how the views worked together in the tool, also done to limit the expected number of interactions from the participant as a form of independent variable control [20]. The final navigational task, defined by A.7, tasked the user with finding the three available corpus navigation methods that were mentioned in Section 4.11. After this, the participant was given the task to navigate to the specific document *A Doll's House* by Henrik Ibsen, with their preferred method of navigation, according to A.8. Similarly to the initial navigation task, these tasks aimed to explore both how easy the alternatives were to find and if a specific method was preferred by the participants. However, since it was a question of familiarity in this specific task, the last method they found was noted down, since this could affect their preferred method [20]. Additionally, they were asked why they picked the specific method they did to collect qualitative data on this alternative in the tool.

With the correct document open, the following two tasks, covered by A.9 and A.10, focused on tasks related to the document-specific views. Firstly, they were tasked with finding the specific word *Money* in the text, also based on control of independent variables, i.e., the same word for every participant [20]. This word was chosen in particular since it did not appear at the beginning of the text, and it only appeared 48 times in the text, meaning that it would not appear directly in the word frequency view either. Every participant was provided

with said occurrence of the word along with the extra task to specifically find the fifth occurrence of the word, where the goal was to get the participants to use both views for this specific task. What the hint also meant to achieve was to see if they would use the dual range filter to find said word. With the word and its specific occurrence found, the participants were then tasked to utilize the word trends view to find how many segments the word was significant to and which of these segments it was considered to be most significant to. When the default settings were used, *Money* was always considered to be significant in three segments, where the highest significance score was always found in the second segment. Important to note about the word trends view task, defined by A.10, was that it aimed to answer if the colour scheme used had enough contrast to differentiate between the different words [57]. It also aimed to answer if repetition of colour could be used to provide the participants with the information that the specific word was significant to several segments.

Moving on from the document-specific views, the two following tasks, defined by A.11 and A.12, focused on the node graph and similarity overview. Here, every test participant was tasked with finding the two documents that were considered to have the highest similarity within the corpus and to find the five specific words that contributed to this similarity the most. Given the default settings, cosine similarity with *TF-IDF* scoring, and that all similarity scores were normalized, the highest similarity of 10 was always found between the *Capital stocks and fixed capital consumption* for the years 2023 and 2024 from the *ONS* in the UK. Additionally, the top five words that contributed to this score were *Capital*, *Net*, *Of*, *The*, and *And*, which thus was the information each participant was tasked to find for these tasks. These tasks aimed to firstly find if the edge sizes of the node graph were distinguishable and if the participants could understand that said sizes correlated to the similarity score [29]. The similarity threshold filter was also hinted at here, and if used, it was noted down since the time to complete the task would increase given processing times. The task for the similarity overview, on the other hand, aimed to evaluate the colour choices in the grids, if they were used to find the top five words, or if the participants instead only relied on the tables to find said words.

With all view-specific tasks completed, the participants were lastly asked to go back to the "main view" and describe what data they believed each view showed and what it aimed to provide to the user, as defined by A.13. This task was motivated by the idea that the participants, at this point in the test, had got a familiarity with the tool and its functions, which hopefully made it easier for them to describe the tool. Since the answers were meant to be given as spoken descriptions, they were mainly qualitative, but they would also be evaluated as quantitative metrics since the descriptions were either correct or false. Lastly, as detailed in A.14 and A.15, the test participants were first asked if they had any additional comments, included in the case that the previous tasks failed to highlight any thoughts from the participants. Since answers again were given as spoken words, this was purely meant as qualitative data collection and could be seen as a way to acquire general points of improvement for the tool. Whether the participants had any additional comments, they were finally asked to complete a questionnaire about their experience and thoughts on the tool, and thanked for their participation and time.

5.4 User test questionnaire

As described at the end of the previous subsection, each participant was asked to complete a questionnaire at the end of the test. The English version of the questionnaire can be found in its entirety in Appendix C which, to note, was the original version, while a Swedish translation also was provided if a given participant rather preferred that. A majority of the answer fields in this questionnaire were formulated as 1–5 Likert-type scales, where a score of 1 meant that the participant strongly disagreed with the statement and a score of 5 instead meant that they strongly agreed with the statement [58]. Utilization of these types of scales

also meant that the corresponding questions had to be formulated both in a way that fit the format and, in this case, *positively* worded. To only use statements positively was chosen since it would lessen the mental effort of the participants and ease the scoring process since no answers would have to be reversed [58]. As can be seen in Appendix C, each question was related to a certain part of the tool, which had been used during the test and contained three mandatory Likert items along with an optimal free-form text field each. The Likert items were formulated to cover *ISO*'s three aspects of usability, i.e., *effectiveness*, *efficiency*, and *satisfaction*, while the contents of each of these items were based on the descriptions provided by Frøkjær et al. [24, 25]. Thus, each set of Likert items aimed to provide a score on the usability of the relevant part of the tool, while the free form text field was included in order to once again collect qualitative data. Important to note here as well, since it was a questionnaire, the participants were meant to answer how *they* felt about the statements; thus, their perceived efficiency could vary a lot from the measured efficiency, for example. Finally, these scores were then plotted as box plots with the *Python* library *matplotlib*, similarly to the plots found in *SUS.tools* by *Mixality*. Given that *SUS.tools* only produced plots for the 10 question system usability scale questionnaire, it did not suit the questionnaire seen in Appendix C [59]. Thus, *matplotlib* was instead used to generate representations of the "component based" usability questionnaire used for the user tests in this thesis.

5.5 Evaluation of visual representations

Given the broad scope of the user test and accompanying questionnaire mentioned in Sections 5.3 and 5.4, the evaluation of their results was split into the groups visual representations and interaction techniques. In these processes, the results from the internal validation of the two groups were also included to give a reason behind the results of the user tests. Highlighted tasks for the evaluation of the tools visual representations, i.e., the content of the views, were A.5 and A.9–A.13. This group of tasks, along with accompanying interview questions, specifically aimed to acquire information on how well the views portrayed their information. Here, measured by the time it took to find said information or on how correct the participants answer was, depending on the task. On the other hand, the questions present in the questionnaire focused almost entirely on the views and how their design improved the usability of the tool. The evaluation of the test results, supported by the internal validation, was done with both **RQ1** and **RQ3** in mind, where the results, as mentioned in their relevant subsections, were both of qualitative and quantitative nature. It is therefore aimed to also answer **RQ1** and **RQ3** on the usability metrics acquired from the evaluation of the visual representations [25].

5.6 Evaluation of interaction techniques

The other part of the evaluation of the tool was, as described in Section 5.5, evaluation of interaction techniques. This aims to provide results on the usability of the interactive elements described in Section 4.11, along with those techniques mentioned throughout the view-specific interaction mentioned throughout Chapter 4. For this part of the evaluation, highlighted tasks that were relevant were found in A.6-A.8, along with the result times for the tasks mentioned in Section 5.5. In addition to this, the questionnaire also featured questions specifically meant to provide data tied to the usability of the interaction techniques present in the tool. These questions were mostly aimed at the efficiency metric, since they were related to perceived time frames in the tool [25]. In addition to this, the aforementioned recorded time frames described in Section 5.3 and possible error rates were included as quantitative data collection; it was also ensured that all test participants were tasked with finding the same items [23]. Qualitative data on the interaction techniques were also collected during the user test through interview questions and the "additional thoughts" answers in the

questionnaire. Similarly to the aim behind the evaluation found in Section 5.5, these forms of data acquisition were designed with **RQ1** in mind, since most of the interactive elements were meant to provide the user with ways to find data within a document or corpus. Thus, the results of the evaluation were also meant to be used to answer **RQ1**.

6 Results

As described in Chapter 5, the tool was tested in three distinct stages, the results of these stages will be presented in this section. Given the approach of mixing quantitative and qualitative methods during the interview and questionnaire stages, as seen in Appendices A and C, the results of these stages will be split according to the method used. Since the preliminary testing, described in Section 5.6, was conducted completely *user free* however, those results will be presented in its own section. Important to note is that all results presented in this chapter are based on the *first* prototype of the tool, since the initial implementation and evaluation stage, defined by **O3**–**O5** in Section 1.5, became longer than initially planned. Based on the following results, a second prototype was, however, developed, which aimed to fix the most prominent faults in the first prototype.

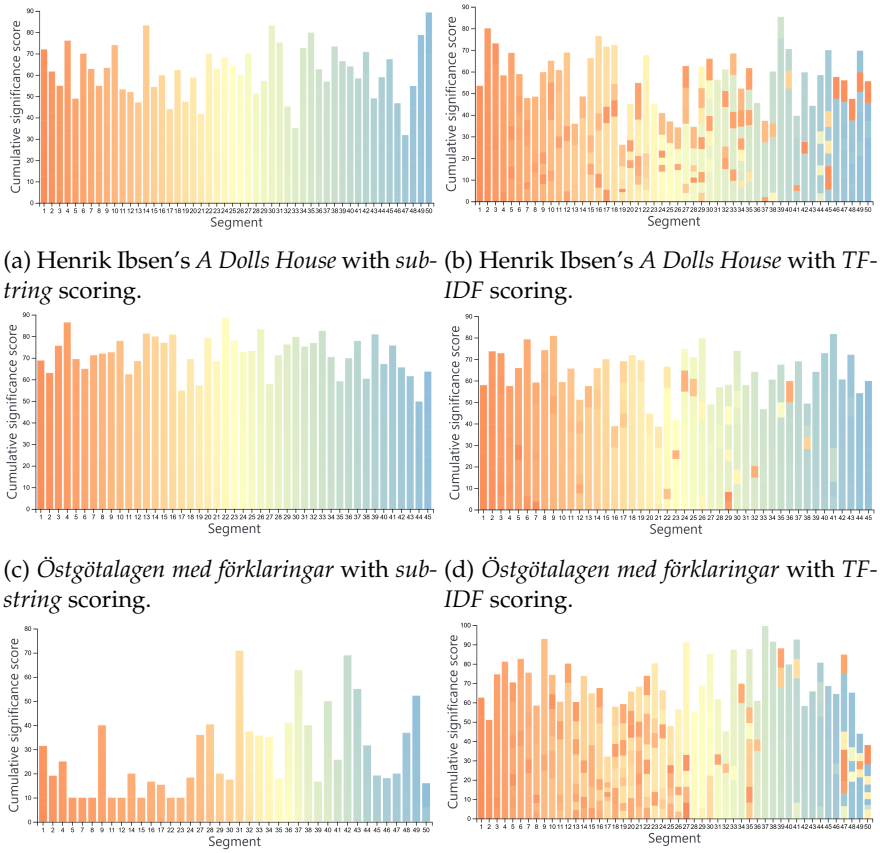
6.1 Preliminary testing results

As described in Section 5.1, internal tests were conducted to acquire results on implemented algorithms and certain design choices within the tool. These tests were kept separate from the user-oriented tests given the formers focus on interaction techniques and visual representations instead.

6.1.1 Implementation of NLP methods

The first part of these tests were in regard to the *regex* filtering methods described in Section 4.2. It was found that it was possible to manually choose what characters to omit from word counting, but the list of `replace()` calls would become increasingly long while it still missed certain non-letter characters. Based on this, it was thus instead chosen to mark all characters, not part of the Latin Unicode character class with a singular call to the `replace()` function. However, to additional `replace()` calls were still included, one to handle shortened words separated by apostrophes in English texts and one to remove longer sequences of spaces. This meant that the tool could process documents written in Latin-based alphabets in accordance to the aim described in Section 1.3.

The two implemented methods for segment-based word scoring, *substring* and *TF-IDF*, did both produce data for the word trends view. However, there was a risk that the *substring* method would skip segments if there was a total lack of significant words within it, a problem



(a) Henrik Ibsen's *A Dolls House* with *substring* scoring.

(b) Henrik Ibsen's *A Dolls House* with *TF-IDF* scoring.

(c) *Östgötalagen med förklaringar* with *substring* scoring.

(d) *Östgötalagen med förklaringar* with *TF-IDF* scoring.

(e) *UK Balance Sheet Estimates 2025* with *substring* scoring.

(f) *UK Balance Sheet Estimates 2025* with *TF-IDF* scoring.

Figure 6.1: Different results of words trends generation with *substring* or *TF-IDF* scoring.

that was not present for the *TF-IDF* method. When loading the user test corpus, mentioned in Section 5.3 and found in Appendix B, the initial view generation, described by Figure 5.1, could on average be completed in a time frame of 11.19 seconds across ten runs, with 45 document segments. The same process with the *substring* method and a substring length of 5 instead had an average completion time of 13.93 seconds. Comparing the resulting visual representations of these methods, shows that the *substring* method usually scores repeated words highly compared to the *TF-IDF* method as can be seen in Figure 6.1. These results were generated with 50 document segments for Figures 6.1a, 6.1b, 6.1e, and 6.1f, while Figures 6.1c and 6.1d instead utilized 45 document segments, all *substring* data was generated with a substring length of 5.

The user also had a choice in method for the similarity evaluation across the corpus, as described by Section 4.2. Loading the user testing corpus with cosine similarity (and *TF-IDF* for segment scoring), the initial view generation was on average completed in 11.19 seconds, since the process was identical to the *TF-IDF* runs. The same process with Jaccard similarity instead could be completed in 18.92 seconds on average. There is also a notable difference between the resulting node graphs from these methods, as can be seen in Figure 6.2, where the default threshold value of 1 was used for link filtering. In the figures it is possible to see that both methods clustered the English documents in one group, but with Jaccard similarity, Figure 6.2b, the Swedish documents within the corpus was not clustered.

(a) The resulting node graph with cosine similarity.

(b) The resulting node graph with Jaccard similarity.

Figure 6.2: Different results of node view generation.

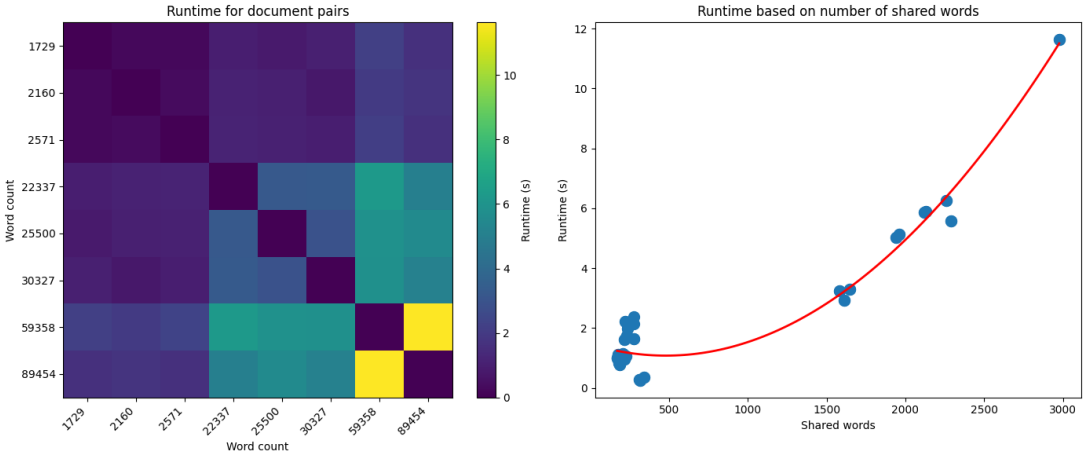


Figure 6.3: Average times to open the similarity overview for documents with $\geqslant 1729$ words. Based on document word count and number of shared words.

6.1.2 Cross-view and corpus navigation

To navigate between views within the tool was mostly instant since the main views were generated during the start-up process, seen in Figure 5.1, with one outlier being the node view which took some additional seconds to handle the force simulation. However, given that the similarity overview, defined as *Sim. Overview* in Figure 5.2, generated its content on the fly, the time it took to open it was largely dependent on the length of the relevant document pair. For document pairs with $\leqslant 354$ words in the test user corpus, the similarity overview could always be opened in less than a second, but the document pairs with $\geqslant 1729$ words did require slightly longer processing times. The average opening times across 5 runs can be seen in Figure 6.3.

In addition to this was the time it took to change what document was shown in the document-specific views, indicated by the *Change displayed document* item in Figure 5.2. The documents with $\leqslant 354$ words could be switched between nearly instantly, but longer documents did require more time to display the information in the different views. Similarly to Figure 6.3, the time to change documents were averaged across 5 runs, with the results presented in Figure 6.4.

6.1.3 Faults with the implementation

Some faults within the tool were found before the user tests took place, but given that they were discovered too late to fix, they were kept in the tool. Firstly, selection of a given word within the close reading view would *always* highlight and scroll the view to the first occur-

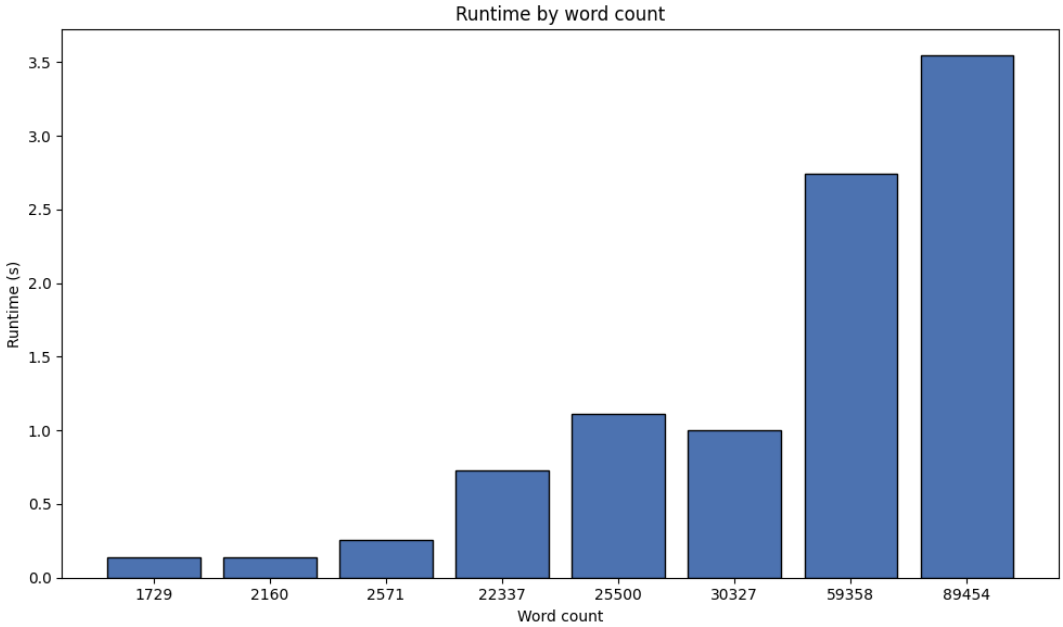
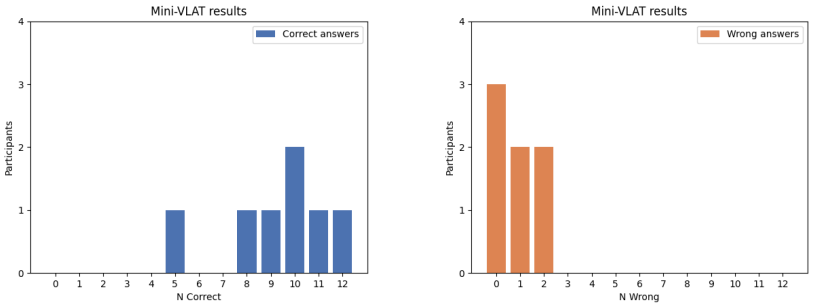


Figure 6.4: Average times to change the displayed document in document-specific views.

rence of said word. For example, if a user clicked on a word in the middle of a document, if the same word was present earlier in the text, the view would scroll away from the word they clicked. Another fault or inconsistency, was that the user had to press the setting bar's confirm button if they wanted to change document through the dropdown menu. Compared to the two other methods, through the corpus overview or node view, this thus required additional interaction to achieve the same goal. A visual fault was found with longer document names where certain elements became larger than intended. In the corpus overview this severely impacted the fingerprints for the documents, sometimes hiding them completely. Headers containing the document name, such as the ones found in the main view and the tables of the similarity overview, were also affected by longer document names where the contents of the header became cramped or flowed outside their boundaries. Lastly, it was discovered before the tests that the nodes of the node view often became to spread out to clearly visualize the indices of all nodes. It was either required to zoom out in the view to see all nodes, thus making the indices unreadable, or to zoom in the view to see the indices, thus hiding some nodes.

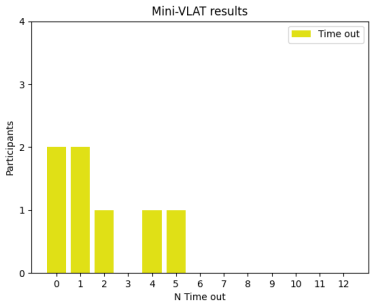
6.2 Test participants

In total, the user test as described in Sections 5.3 and 5.4, was conducted with 7 participants. A majority of the test participants were between 23–25 years old and were in the process of completing their master's theses in the general area of media technology. Apart from purely students, a data scientist in their early 40s also took part of the tests, meaning that all participants were part of the target user base described in Section 3.1. It should, however, be mentioned that the tool was not tested with the entire target user base, for example, no participants held an occupation that handled economic reports, meaning that additional testing would be preferred. Additionally, a majority of the test participants described their English proficiency as "good" and when asked if it more accurately could be described as "fluent", all agreed. Important to note is that one participant gave the more detailed description of their English being on a C1 level, indicating fluency as well [60]. As seen in Appendix A and



(a) Correct answers.

(b) Wrong answers.



(c) Time outs.

Figure 6.5: All answers from the *Mini-VLAT* assessment test.

discussed in Section 5.3, the *Mini-VLAT* assessment test were completed at the beginning of each test and the collected results of that part can be seen in the sub figures of Figure 6.5.

As can be seen in Figure 6.5a, given that it was a 12-item test, a majority of participants got at least $\approx$ 67% of the answers correct, but there was one outlier who answered correctly on $\approx$ 42% of the questions which should be noted. As indicated by Figure 6.5b, there were no participants that had a failure rate above $\approx$ 17%. The number of questions which the participants timed out on is mostly the same as the outright wrong answers, but with two participants who timed out on either 4 or 5 questions, as seen in Figure 6.5c.

6.3 Quantitative user test results

Given the described structure of the user involved tests in Sections 5.3 and 5.4, quantitative results were collected as timing data for relevant tasks and error rates during the interviews, and through Likert-type scale answers in the questionnaire. Additionally, as described in Section 5.4, the questionnaire results were plotted as box plots with *matplotlib*, and this structure is seen in Figure 6.6, where each component has been given its own sub-plot. As can be seen in the plots, most of the components achieved median scores between 4–5 on perceived effectiveness, efficiency, and satisfaction. Similarly, the efficiency Likert-item for the node graph view related to if the colour of the nodes helped to lessen the time taken to find similarity scores between documents. Apart from this item, the result is thus that a majority of statements from the questionnaire were statements that the participants agreed with.

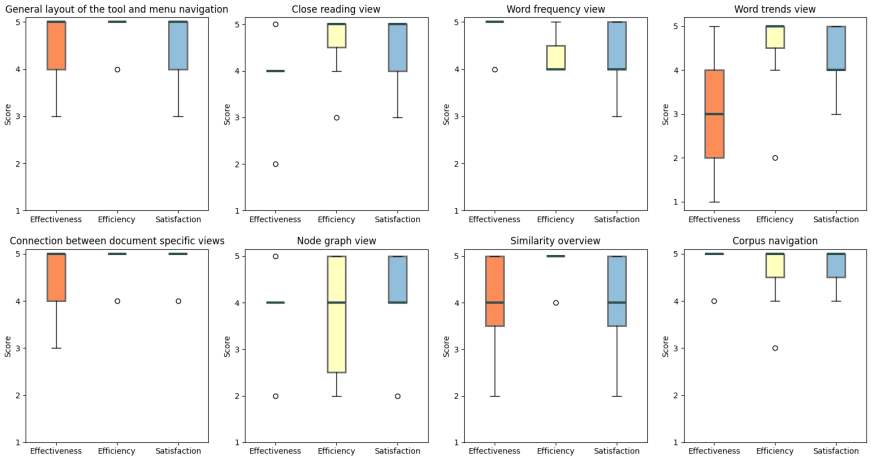


Figure 6.6: The questionnaire answers divided by component with each Likert-item plotted separately.

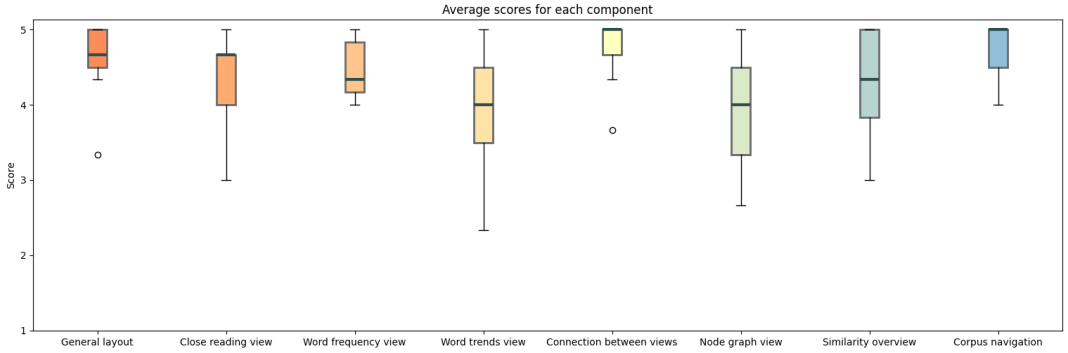


Figure 6.7: The questionnaire answers divided by component with average scores for each component.

To further find an overall score for each component, the three Likert-item answers were averaged for each participant, which is presented in Figure 6.7. This average data clearly shows the interquartile range for each box, compared to Figure 6.6, where it can be seen that the first, third, fifth, and eighth generally has low ranges. The second, fourth, sixth, and seventh components had noticeably larger ranges instead. Additionally, in this figure, it can be seen that all medians lie between 4 and 5, with some lower outliers or minimum values outside the “agree” range.

As mentioned at the beginning of this section, the interview stage did feature time dependant tasks, where the results can be seen in Figure 6.8.

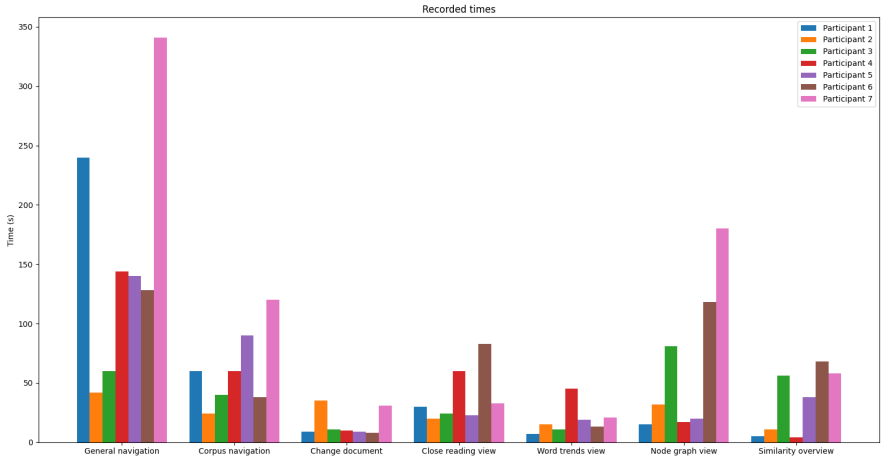


Figure 6.8: The recorded times for each participant on time dependant tasks.

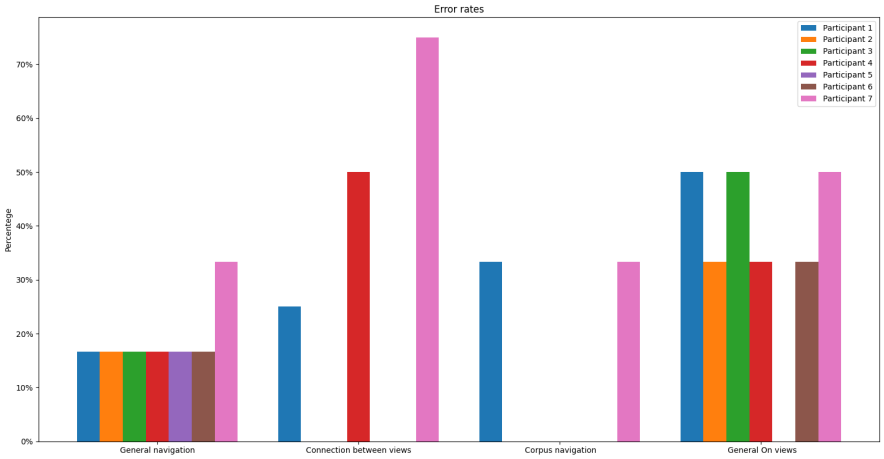


Figure 6.9: The error rates for tasks where it was applicable.

Here it can be seen that the task with the highest recorded times overall was the first one. For the other five tasks the recorded times were on average lower, only featuring a few outliers who either was affected by slow processing in the tool or since they decided to explore some other features of the tool. Regarding the first task, no participant could find and identify all views in the tool without a hint, meaning that each participant had an error rate of $\approx 17\%$ or above on this task. Other error rates can be seen in Figure 6.9, where the first and third tasks were navigational tasks while the second and fourth task instead focused on whether the participants could describe the functionalities of the tool correctly. As can be seen in the plot, a majority of the errors appeared during the last relevant task, caused by wrong descriptions or a lack of descriptions for certain views. This is also the cause behind the three spikes in the second task.

6.4 Qualitative user test results

Given the descriptions of the two user-oriented stages, found in Sections 5.3 and 5.4, a majority of the qualitative data was collected during the interview stage, in the form of written notes. These notes were taken either from discussion between the interviewer and test participants or from ideas and thoughts the test participants spoke out loud. The other group of qualitative results were collected from the questionnaire's open form answers, found in connection to each question as seen in Appendix C.

6.4.1 General navigation and interface layout

To begin, following the order of the tasks found in Appendix A, are the results related to the general navigation within the tool and the placement of different views and buttons. Based on the acquired qualitative results from this area of the tool, there was a general sentiment that the layout as a whole was well-defined, clean, and followed a consistent structure. Thus suggesting that the recorded times for this task, seen in Figure 6.8, could be caused by the multiple steps of the task instead of difficulty. This in turn provided the test participants with an "intuitive" experience of the general user interface. There were however some more negative sentiments expressed as well in regard to this task, for example a problem encountered by one of the test participants where the node view was zoomed in on startup, in turn making the experience confusing since it was difficult to decide whether the view worked or not. Other general improvements for the tool as a whole included implementation of search functions for several views, to avoid use of the `ctrl+f` function present in *Google Chrome*. Relevant to general improvements were also feedback left by two participants about providing the user with a way to scale the sizes of the views as they desired, one participant also explained that it was preferred to have most views visible simultaneously. This further explanation was based in earlier experiences with visualization tools that usually provided singular views at a time, thus making comparison between different data encoding methods harder. In addition to this, there was one participant which suggested that the width of the close reading view could be shrunk to 50%–60% of its current size to provide more screen space to the more "data centric" views. One problem that most of the test participants encountered but only one explicitly mentioned was how slow certain features of the tool were. This comment mostly related to the filtering method present in the frequency view, but was also extended to the process of changing documents through corpus navigation and opening the similarity overview.

6.4.2 Corpus navigation

Regarding corpus navigation, covered by A.7, A.8 and the final question in Appendix C, there were the positive sentiments that the method was simple in design and inclusion of several ways to change documents was an appreciated choice. When questioned on their preferred way of navigation, the test participants gave several different answers, mirroring the fact that all methods were used by at least one participant during the tests. Some participants preferred the dropdown menu compared to the other ways since they were provided with the name of the document and could quickly find said name within that menu. On the other hand, the way featured in the corpus overview was preferred since the name could be searched with `ctrl+f` to quickly find it, one user also stating that they preferred it over the dropdown view since they did not know how that view was sorted. The corpus overview was also chosen by one participant, not since they preferred the method but instead given that it was the method they assumed "general users" would prefer. They instead expressed that their own preferred method was the dropdown menu from a "data perspective". The sole test participant who used the node view to navigate to the specified document did so since they had identified the cluster of Swedish documents. Thus, they knew the English document

would be present in the other cluster and decided to search there, this process was, however, done without the knowledge of what the numbers in the nodes meant. As previously stated, one large problem noted with this functionality was the significant slowdown when the displayed document was changed, where the cause of frustration was directed towards the lack of feedback for this action.

6.4.3 Document specific views

Moving on to the document-specific views, the answers were more direct, either focusing on prominent faults within them or highlighting certain features that elevated the usability of the view. There were few outstanding features brought up related to the close reading which elevated the usability of it, but all test participants expressed that they understood that the view was mainly used to provide the displayed document as plain text. What the feedback regarding this view focused on instead was that the segment breaks, and especially their colours, were difficult to understand and that explanations of the functionaries within the view would be preferred for a more effective and satisfying experience. Additionally, given buggy implementation in the tool, a cause of frustration within this view was experienced when trying to select a specific occurrence of a given word, which always would select the *first* occurrence of said word. This thus lowered the expressed satisfaction when utilizing this functionality.

Similarly to the close reading view, a majority of the expressed positives with the frequency view was that its intended uses were easy to understand. Areas for improvement were instead highlighted as incorporation of a search function and number input for the range slide, while some participants also pointed out that the most frequent words usually were the least interesting. In addition to this, the arrows used to click through occurrences were highlighted by one participant as a good feature, but it would be preferred if the tool showed that they were clickable. The word trends view, compared to the previously mentioned document-specific views, got significantly worse feedback. Most of the negative sentiments were related to how little information was provided about the colour palette used and what *segments* meant, while one test participant also noted that it was difficult to understand what was meant with *trends*. There was however one outlier among these answers who expressed that the significance scores were well established. However, this participant still expressed that it was difficult to understand the difference between the occurrence of a specific word and the number of segments it was considered significant to. However, a majority of the test participants noted that they were able to find the scores and segments which were relevant to a selected word. In addition to this, a majority of the participants expressed that they understood the highlighting functionalities across the document-specific views, with one participant specifically describing that it was a good feature to include in the tool.

6.4.4 Node graph view and similarity overview

In regard to the node graph, apart from the navigational functionality, the most prevalent sentiment was that the tool did not provide enough information about the links being clickable. In addition to this, the view was generally described by the test participants as being difficult to understand, some citing that they either did not know what the colours meant or that a cluster had been missed due to the zoom and pan functionalities. Another point of confusion highlighted by one of the test participants was that links of similar strengths (for example 9,4 and 10) were difficult to distinguish between, a problem found in more dense clusters as well, given the overlapping links. What a majority of the test participants expressed that they understood, however, was that the view showed how documents in the corpus were connected. One point of improvement that was also suggested in regard to the node graph view was the gravity between nodes, as it currently stood, it was easy to lose track of dif-

ferent clusters given that they were spaced out too far from one another. The feedback on the similarity overview was generally more positive compared to the node view, only having two participants express that they did not understand what the word grids represented. The tables were, however, expressed to be easier to understand and generally preferred for information searching among a majority of test participants, but, as with feedback on other views, a dedicated search function would be a welcomed feature to search in the tables. Two participants also noted that the grid representation would be more useful with additional interactive techniques, such as the squares being clickable to highlight the relevant words within the tables.

6.4.5 Summarization of qualitative results

To summarize the qualitative answers, and present feedback on the tool as a whole, there was a consensus from the test participants that there should be additional information provided about the different views if needed. Shortened words, such as those found in the tables of the similarity view, was also expressed by several participants to not provide a sufficient amount of information. One participant noted that they would prefer a tool tip to show the entire word, which according to them would solve that problem. The general colour usage should also be reconsidered, or explained with colour legends since it was expressed by several participants that they did not know what certain colours meant in different parts of the tool. The use of similar colour palettes across different views was also a point of confusion, where one participant expressed that they perceived a connection between certain items which were not connected.

7 Discussion

The previously described implementation and evaluation phases of the thesis work along with the results of these stages will be analysed and discussed in this chapter. This analysis and discussion will handle both faults and strengths found within the previous chapters and explain them based on the results. In addition to this, the analysis and discussion will be connected to the research questions found in Section 1.4.

7.1 Method

Based on the defined objectives found in Section 1.5, it was chosen to plan the thesis work according to Munzner's nested design model, seen in Figure 3.1 [27]. Even though it was not followed exactly, it did provide a good foundation to implement the tool on. Since both **O7** and **O8** were mostly skipped, as indicated by the introduction of Chapter 6, the tool was not implemented iteratively. This approach to implementation was mentioned in Section 3.5 and was a central part of the nested model which was skipped during the thesis work.

7.1.1 Planned design

The aforementioned nested model was still followed closely during the planning of how the tool would be designed, covered throughout Chapter 3. Given the aim to provide a text visualization tool which was both generalized and lightweight, it was possible to characterise a possible domain problem that ended up being followed throughout the thesis work. Further on, based on the objectives proposed by Wehrend and Lewis, specific goals which the user would have the possibility to achieve with the tool were defined [28]. Based on the selected objectives along with definitions on possible data types, suitable visualization techniques could be chosen. One part of Section 3.2 that was not followed in the implemented tool, however, was character positions across the corpora, since it was decided to keep document and corpora-specific views separate. The planned file support, described in Section 3.3, was implemented even though this functionality was not covered by the preliminary testing or the user test. Inclusion of this support did, however, allow for different types of documents to be used during these testing phases, ensuring that the sources mentioned in Section 3.1 was part of the tests. To consider several different visualization techniques to be tested against one another was suitable according to **RQ1**, but as indicated by the introduction of Chapter

6, these plans were not followed given time constraints. This lack of comparative testing was also extended to the alternative layouts, indicating that the initial design plan did contain parts which were too ambitious or wider in scope than indicated by the aim of the thesis.

One large difference between the planned design and the implemented tool was the considered approach to utilize *Python* libraries for file conversion. All described file formats in Section 3.4 was still supported by the final tool, but purely through the use of native *JavaScript* functions and the libraries *PDF.js* and *JSZip*. Thus ensuring that the tool remained front end only, in accordance with the aim of Section 1.3, with the added benefit of relying on fewer external libraries. What Section 3.4 also failed to cover was the inclusion of several *NLP* methods which later were included in the tool. Apart from the mentioned differences between plan and final tool, the nested model was useful to define what needed to be implemented to fulfil the aim of the thesis while taking the delimitations into account. Lastly, the described evaluation in Section 3.5, though not being followed fully, was later used to define the different parts of the evaluation process in Chapter 5.

7.1.2 NLP methods

As described in Section 4.2, the implemented *substring* method was only based on the n-grams method proposed by Cohen, since the method description was misunderstood. The three other analysis methods did, however, work as expected, but inclusion of even more methods would be nice to provide as many options as feasible to the user. The other part covered by Section 4.2 was word extraction. The chosen method to filter out non-letter characters with regular expressions was suitable and worked well for Latin-based alphabets, based on testing with English, Swedish, Italian, and German. However, the choice to ignore the full forms of shortened words and split them at the apostrophe, meant that the words weren't accounted for correctly by the *NLP* methods. This was, however, chosen as the preferred approach since the conversion to the full forms of the words would be language dependant, meaning the *NLP* methods in the tool would be less generalized.

7.1.3 Visualization methods

Implementation of the different visualization methods was mostly done, as indicated by the delimitation found in Section 1.6, with *D3.js*. The planned encoding approaches of utilizing a vertical bar chart for the frequency view and a node graph to visualize the corpus, as described in Section 3.3, was kept throughout the entire implementation process.

The stacked bar chart approach in the word trends view, however, was first considered to be done as a line plot instead where the significance of all words would rise and fall across a given document, similarly to the trends view found in *Voyant Tools* [3]. However, since this would require large number of lines, coloured differently enough to be distinguishable from one another which would allow for around 12 different lines [57]. Thus, the implemented method was chosen instead, which interpolated between 3–4 colours across the segments of the text, assigning a word with the colour of the first segment which it was considered to be significant in. The aim here was thus to show if certain words stayed relevant across the text and if this significance increased or decreased throughout it. Since it was found that the *TF-IDF* method produced more unique charts as well, as described in Section 6.1.1, it was chosen to use this bar chart for fingerprinting of the documents as well. It was figured, based on the work by Keim and Oelke, that this would give the users a quick reference image to identify the documents by [50].

Within the similarity overview, inspiration from Keim and Oelke was taken once again when implementing the grid-based document representations featured in the left and right columns. This was not meant to be fingerprints, however, but instead meant to visualize every unique word in each document and to give the user a quick way to see if a majority of words was shared between the documents.

Lastly, the close reading view did not feature many visualization methods, mostly relying on the highlighting features, but with segment breaks included. These aimed to provide a way for the user to see which words were part of what segment for the word trends analysis methods. To further highlight this connection they also shared the colours of the relevant segments.

7.1.4 Evaluation

As described in Chapter 5, it was chosen to split the process into three distinct steps, one user-free and two with users participating. This approach proved useful since it was possible to test how well the tool handled and processed files independently of how users interacted with the tool and their perceived experience with it.

Starting off, it must be stated that the validation step during the implementation phase and preliminary testing was not well enough structured to provide an efficient workflow. This process boiled down to notes taken during implementation about things that either were slow, didn't work as expected, or if the chosen methods felt suitable. It was still possible to acquire results with this approach, but if planned more in depth it would be a smoother process.

Evaluation of the user test results, on the other hand, was planned in greater detail given the definitions of which tasks focused on what areas of the tool. This part of the evaluation process was also planned to strengthen certain points found during the user-free testing as well. Thus, the weaknesses during the preliminary tests and the validation of the tool were planned to be avoided with this approach. In addition to this, the evaluation of the results were planned to lay a foundation which could be used to answer the research questions found in Section 1.4, as described in Sections 5.5 and 5.6.

The descriptions found in Sections 5.3, 5.4, 5.5, and 5.6 handled the general structure of the user tests along with how the results of them would be evaluated. These areas of evaluation did however not cover the question of internal and external validity, aimed to answer how the evaluation process was balanced, along with the goal to find out if the results of the user tests were enough for analysis [19]. Based on the evaluation patterns presented by Elmqvist and Yi, the question could be answered through, for example, *pilot studies*, *coding calibration*, *prototypes*, or *statistic verification* [19]. As seen in Objectives **O5**–**O8** from Section 1.5, the goal was to evaluate the tool as an iterative process, where each version of the tool could be classed as different prototypes in this context. This certain pattern was included since it aimed to collect formative feedback on the early versions of the tool, which could identify potential problems, either to avoid faulty implementations or to aid the evaluation of the tool overall. One part of the evaluation process which aimed to answer the question of validity, was of course the process described in Section 5.1, which in this context was considered to be a form of coding calibration. It was thus included to, apart from the use of it in the user test evaluation, serve as a form of resource saving measure [19]. On the question of validity of the evaluation process, how data was collected along with what this data was also had to be accounted for. This involved investigation of the necessity of certain data which was collected during the user tests, or included tasks in the test, which referred to whether the data was usable [19]. During user tests for example, it was chosen to not include information or questions related to the test participants gender since it was omitted from the target audience, as seen in Section 3.1, and given that their gender was figured to not have an effect on how useful the tool was to them. Additionally, as mentioned in the preceding sections, the same corpus and data was tested with all participants to produce similar visual representations every time as a form of factor control. These validity measures, along with the Mini-VLAT assessment to ensure all participants were part of the target audience, thus aimed to answer the question of validity for the test data.

Along with questions about the validity of the evaluation, there was the question about its reproducibility. This referred to how the evaluation process was being conducted to ob-

tain consistent measurements and results, with identical input data [61]. This approach was reflected in the preceding sections, were it was highlighted that test users utilized the same corpus and completed the same tasks. Done to ensure that the input data stayed consistent.

7.1.5 User test design

The two parts of the user tests, seen in Appendices A and C, was designed to test specific parts of the tool while collecting both quantitative and qualitative data. The design of these parts were done with statements from Section 3.5 taken into account, i.e., the problems related to validity and incorporation of tasks that focused on user feelings, completion times, and degrees of accuracy [20, 27].

For the interview part, it was decided to include reasoning behind each task which proved beneficial since made the script writing easier while also defining what data would be collected for each task. There are however some mistakes within Appendix A, such as describing the entire test as quantitative in A.5, which has been chosen to be included to provide the exact script which was used during the tests. Another fault with the design of the interviews was lack of rigid instructions to the participants regarding what they could or couldn't do during a task. This was made apparent when the results were compiled, where it was found that task completion times varied a lot as seen in Figure 6.8.

The questionnaire part, on the other hand, was designed in a component-based manner, aimed to find how usable the participants felt the different parts of the tool were. Here the usability was defined as effectiveness, efficiency, and satisfaction, all scored on 1–5 Likert-type scales [25]. The initial reasoning behind the utilization of Likert-type scales was to compile the results with *SUS.tools*, but since the questionnaire was designed before ensuring what data *SUS.tools* supported, it was found that this approach was not suitable [59]. Since the questionnaire was not in the form of the System Usability Scale, the data was instead complied with *matplotlib*. This approach did, however, make it possible to evaluate the usability of specific views or functionalities individually, something that helped when revising the tool.

7.2 Ethical discussion

Since the tool, according to Section 1.3, was meant to be deployed as a web application, there was the requirement to acknowledge how personal data should be processed if handled by the tool. According to the *General Data Protection Regulation*, enforced in the European Union, each user would have to consent to the handling and processing of their data [62]. Given that the tool was able to handle user submitted content, a window, alert, or pop-up was suitable to implement to notify the user about the processing of personal details. This was preferably be the first thing opened when the tool was accessed, which allowed the user to consent to the processing, along with a provided description of how the process worked. However, in the context of user test, with a predefined corpus, this confirmation window was not required since no personal information would never be submitted to the tool. In addition to this, given the aim of a front end only implementation as described in Section 1.3, user data would never be stored on the back end and thus there was no need to include said consent form.

Apart from the ethical concerns that related to how personal information was handled, there was the question if the data was represented with an ethical approach. This referred to the requirement to acknowledge that there was a risk that representations either were misinterpreted, or oversimplified, that could lead to incorrect conclusions [63]. This was accounted for during the development process (design, implementation, and evaluation), through internal validation, as described in Section 5.1. It was also accounted for during the user tests to figure out if the representations were misinterpreted, which would signify that something had to be changed in how the views were drawn.

7.3 Results

Comparison between the preliminary testing, quantitative, and qualitative results was used to evaluate the usability of different parts of the tool, as described in Section 5.5. This is done with the idea that certain quantitative results can be explained based on the internal testing and qualitative results. Here, the former is used to explain certain time requirements or usability scores while the latter is used to validate the usability scores. In addition to this, the internal testing and qualitative results was used to define specific parts of the tool that could be improved. These results will be discussed in the order of the Subsections found in Section 6.4.

7.3.1 General navigation and interface layout

As indicated by the usability scores for the general navigation interface layout, the participants generally agreed with the statements found in Appendix C, even though it was the task with the highest completion times. As mentioned in Section 6.4 however, some of these times was caused by the user interacting with different parts of the tool instead of completing the task. Thus, this was more a fault with the task rather than the tool, which also was strengthened by the qualitative feedback of the layout being clean, with a consistent structure, and intuitive.

In the context of general navigation, interface layout and the tool and the tool as a whole, several improvement ideas were provided where some could be connected to both **RQ1** and **RQ3**. Firstly, it was pointed out that more space could be given to the document specific views by shrinking the size of the close reading view since it contained a lot of unused space. Another, layout-centric change was to allow the users to scale the views to their liking. Given that these improvements were connected to the layout of the tool, they were suitable qualitative results for **RQ3** and were also considered for the revised tool. Text based filtering methods in several parts of the tool were also discussed to be a good point of improvement for the tool. This would provide a more specific alterative to *Google Chrome's* `ctrl+f` function while also helping the user to find the locations of desired words within a given text, i.e., a method relevant to **RQ1**, though not a visualization method.

7.3.2 Corpus navigation

Similarly to the results from the previous sections, the average usability score for the corpus navigation functionalities indicated that the participants generally agreed with the accompanying statements, as can be seen in Figures 6.6 and 6.7. Compared to the other recorded times, corpus navigation task was generally completed within a longer time frame since some participant did not know they were allowed to open and interact with the settings bar. This was a fault with the script since it either was mentioned that the settings bar wouldn't be used during the test, or that things shouldn't be changed within it, thus lessening the participants incentive to interact with the menu. Once all methods were found, however, it was commented that the inclusion of multiple ways of navigation was appreciated, but most of the participants did choose to utilize the way featured in the corpus overview. What the corpus overview lacked however was the previously mentioned text based filtering method on document titles and as indicated by Figure 6.4, larger documents did slow down the change functionality.

The task to change to a specific document was, however, done rather quickly as seen in Figure 6.8, where the fastest participant utilized the `ctrl+f` function. This indicated that it could be solved quickly if all participants could see that it was a valid option through inclusion of text based filtering.

7.3.3 Document specific views

The document-specific views, i.e., close reading, word frequency, and word trends view, were evaluated with a combined close reading and frequency view task and a separate word trends view task, based on the selection from the previous task. Based on the quantitative tests, these three views were considered less useful compared to the two previous components, but as seen in Figure 6.7, they were still considered usable by a majority of the participants.

The biggest fault with the document-specific views and the reason behind their lower usability scores were the chosen colour palette, or more specifically the lack of information about the meaning of the colours. This is also highlighted by the lower effectiveness of the word trends views box plot in Figure 6.6, which was linked to the colour usage in that view. On the other hand, though, task completion times and error rates were low for a majority of the test participants during these tests, meaning that it was possible to find specific information with the provided methods. One improvement for the revised tool was thus to include information about each view so that the users quickly could check what the information was displayed in the views, along with a colour legend for the word trends view. Another improvement was to change the colour palettes for the word frequency and word trends view so they didn't share main colours, based on feedback provided by one of the participants.

7.3.4 Node graph view and similarity overview

Similarly to the document-specific view, the node view and similarity overview did receive lower usability scores compared to the higher scoring components. As indicated by the qualitative feedback in Section 6.4.4, these scores were caused by the lack of information, either about the colour usage or the click functionalities of the edges. It was also found that edges were hard to differentiate between, being another reason behind the lower scores, and higher time frames. This was also considered to be revised through inclusion of written descriptions about the view and a smaller drop shadow could also be added along the edges for differentiation. Based on the preliminary testing results about node gravity, described in Section 6.1.3, it was also considered revising the node gravity and edge lengths to position possible clusters and detached nodes closer together. This problem would also partly be solved by increasing the size of the node view.

Given the box plots for the similarity overview and generally lower completion times compared to the node view, the similarity overview was generally more useful to complete the provided tasks. Since the tables, and accompanying sorting methods were expressed as being preferred methods for information searching they would not need to be revised. On the other hand, the colour usage would need to be described, along with providing information on what the grid representations meant would need to be included based on the feedback in Section 6.4.4. As with the word trends view, one revision would be to include colour legends for the three main colours in the view, and to add an information box to explain the view as a whole. Lastly, based on feedback on the intractability with the grid view, another revision would be to include click functionalities to each square in the grid which would scroll relevant tables to the corresponding word.

7.4 Follow-up implementation refinements

Based on the discussed results and possible revisions in Section 7.3, a second version of the tool was implemented. The main view of this tool can be seen in Figure 7.1, where the user test corpus found in Appendix B has been loaded. Some changes compared to the main view seen in Figure 4.12, should be possible to see immediately, but each revision category will be further explained in the forthcoming subsections.

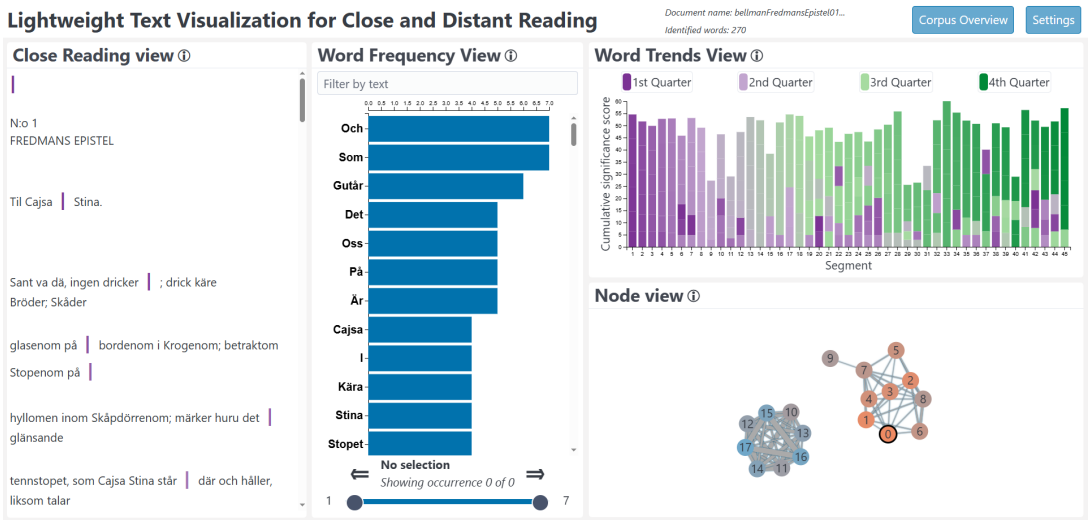


Figure 7.1: Main view of the second version of the tool.

7.4.1 Changes to colour palettes

As can be seen in Figure 7.1, the word frequency view and word trends view now feature distinct colour palettes, based on the feedback discussed in Section 7.3.3.

Here it was chosen to utilize the same blue colour found in the occurrence filter slider for the bars in word frequency view. Apart from differentiating the data from the word trends view, this also aimed to make a visual connection between the filtering method and displayed data through the use of colour.

For the word trends view, it was instead chosen to utilize a colour palette with four colours, thus making the interpolation ranges smaller between segment. This diverging colour palette was provided by *ColorBrewer* which also ensured it was colour-blind safe [48]. To still keep the visual connection between the close reading view and the word trends view, this palette was also used for the segment breaks. Since the corpus overview featured fingerprints encoded with the same method as the word trends view, the same palette was also used there.

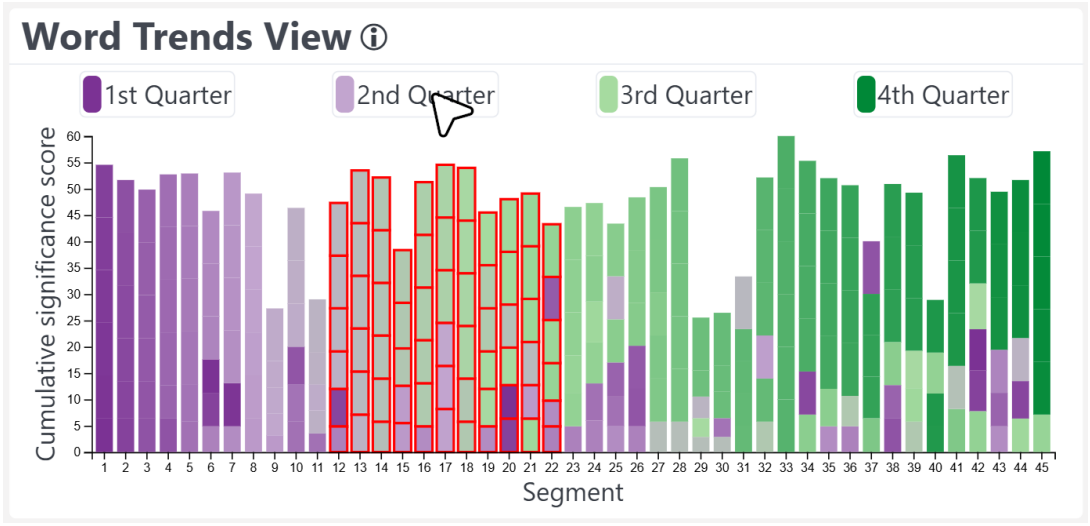
7.4.2 Added functionalities

This revised version of the tool also featured new functionalities, one being the text based filtering method seen in the word frequency view in Figure 7.1, as discussed in Section 7.3.1. Given the discussed revisions in Section 7.3.2, a similar filtering method was also included in the corpus overview, but filtering by document title instead of words.

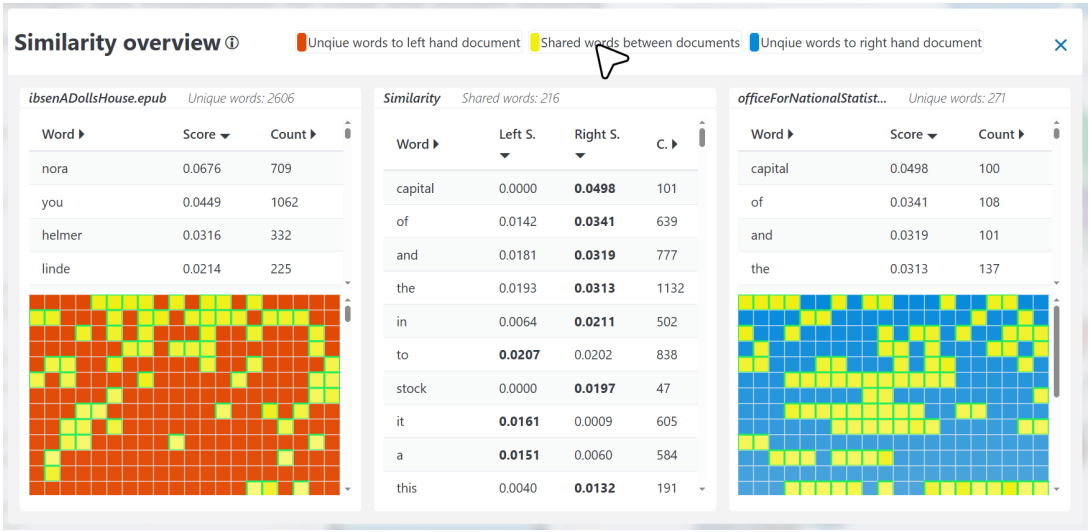
Another new functionality was the addition of colour legends which could be used for highlighting of relevant data groups. While the legend can be seen in the word trends view in Figure 7.1, a revision mentioned in Section 7.3.3, the functionality was also used in the similarity overview to describe what the colour of the squares meant. The highlighting functionalities within this view are shown in Figure 7.2.

Within the similarity overview, a selection function was also implemented, based on the discussed revisions in Section 7.3.4. This made it possible to click any square within the grid view and the tables which contained the relevant word would be scrolled to show it. In addition to this it was also chosen to colour the selection in green, mimicking the selection function seen in Figure 4.4, as can be seen in Figure 7.3

A bug connected to the close reading, described in Section 6.1.3 view was also fixed, now making it possible for the user to click on a specific occurrence in the document and having that specific occurrence selected.



(a) Legend highlighting in the word trends view.



(b) Legend highlighting in the similarity overview.

Figure 7.2: Legend highlighting featured in the second version of the tool.

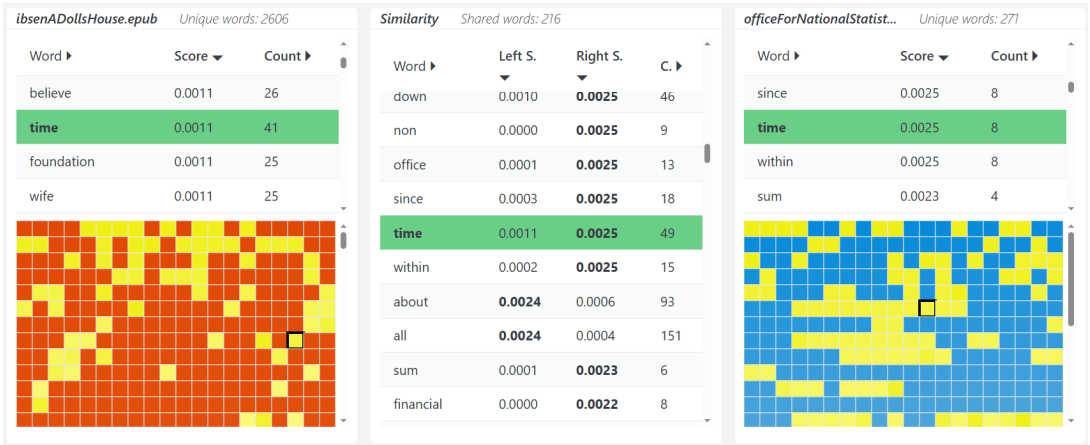


Figure 7.3: Selection of a given word within the similarity overview.

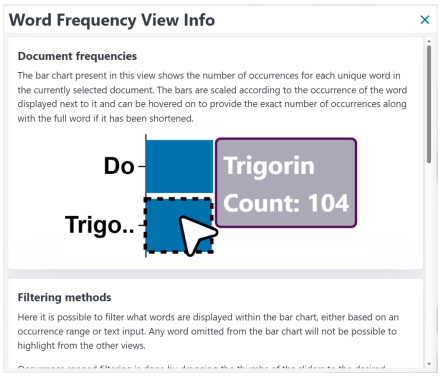


Figure 7.4: Information box for the word frequency view.

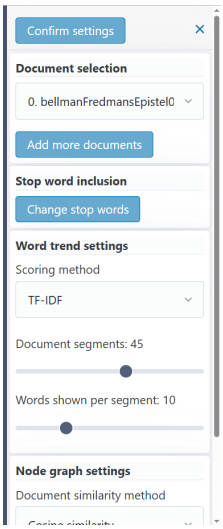


Figure 7.5: Revised settings bar.

7.4.3 Added information

Based on one of the most reoccurring faults brought up in Section 7.3, that being the lack of information, it was chosen to add information to each of the six views within the tool (close reading, word frequency, node, similarity overview, and corpus overview). These were available from the info button found in the header of each one of these views and once clicked they would open a dialogue window with descriptions and images relevant to the view. As indicated by the discussions in Section 7.3, the aim of these information boxes was to provide additional information which the views couldn't provide "intuitively" to the target user based described in Section 3.1. In Figure 7.4, the information box for the word frequency view is shown as an example. The other information boxes follow a similar structure with cards indicating different categories of information and with images featured below relevant paragraphs.

7.4.4 Layout changes

As can be seen in Figure 7.1, it was chosen to decrease the width of the close reading view by the amount described in Section 6.4.1. This made it possible to increase the widths of both the word trends and node views by the same amount, thus providing more space to the visualization methods within those views. It was, however, chosen to also revise the layout of the settings bar, firstly moving the confirmation button to the top to the top of the field to show the user that it was present. The other revision focused on the different settings classes which were collected with different cards, aimed to mimic the design found in the rest of the tool. This revised settings bar can be seen in Figure 7.5.

7.4.5 Node view

Given the discussed revisions in Section 7.3.4, the gravity and edge lengths of the *D3.js* simulation was readjusted, which put node clusters closer together. This also ensured that detached nodes would not stay too far away from the centre of the view. Additionally, based on feedback regarding overlapping edges in Section 6.4.4, it was chosen to include a small drop shadow along each edge which would properly show the edge crossings. In Figure 7.6, this revised node view is shown with the user test corpus loaded where cosine similarity has been used as the analysis method and a similarity threshold of 2 has been applied. As can be seen in the Figure, it is now possible to zoom further in on the nodes while still keeping every

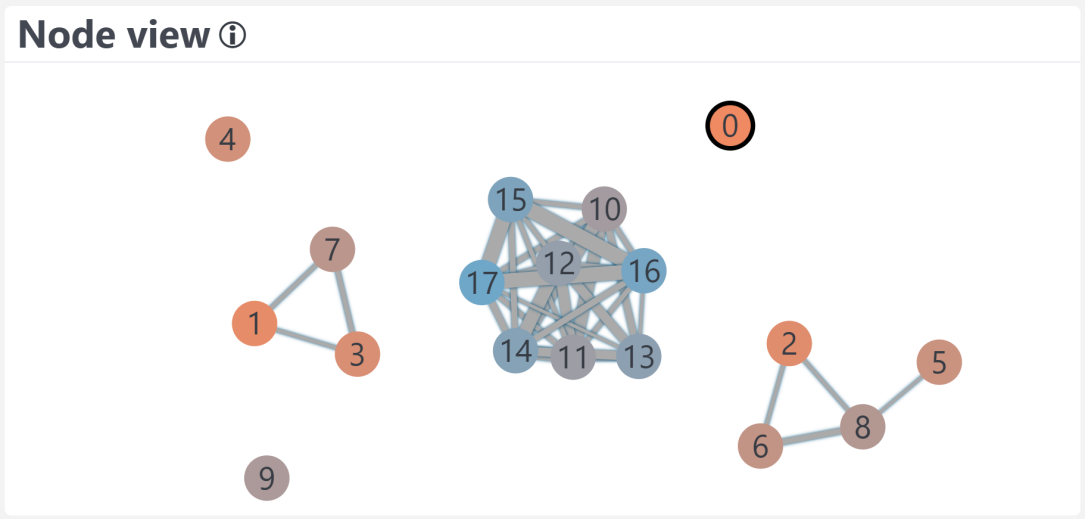


Figure 7.6: Revised node view with the corpus found in Appendix B loaded.

node of the corpus visible. Compared to the previous simulation settings, seen in Figure 6.2 all the indices can now clearly be read and in addition, it is possible to see each individual edge. Thus addressing a prominent problem with the node view as discussed in Section 7.3.4.

8 Conclusion

Within this chapter, the thesis work will be summarized in addition to comparing the Objectives found in Section 1.5 with the *actual* workflow of the thesis. Additionally, the findings of the user tests along with the discussion about them will be used to answer the Research Questions found in Section 1.4. Lastly, some discussion about possible areas of future work for the *Lightweight TextVis* tool will be discussed, presenting functionalities which were not implemented and further evaluation which would benefit the tool.

8.1 Thesis summary

Given the aim of the thesis to explore how a lightweight web-based text visualisation tool to provide users with generalized document and corpus analysis methods with few assumptions on user inputs, two prototypes of the tool *Lightweight TextVis* was implemented. Both prototypes had support for left-to-right Latin-based alphabets and text extraction support for the file formats *TXT*, *HTML*, *XML*, *PDF*, *EPUB*, *ODT*, and *DOCX*. Based on the aim and limitations found in Sections 1.3 and 1.6, *Lightweight TextVis* was implemented as a web-application in JavaScript, largely dependent on the library *D3.js* to visualize data extracted from provided corpora. The first prototype of the tool was tested with users part of the target user base described in Section 3.1, in accordance to the user-based delimitations found in Section 1.6. Based on discussions about the results of these tests, the second prototype was developed, covered by Section 7.4, to handle feedback of and found faults in the first prototype.

The Objectives for the thesis, presented in Section 1.5, were created based on the Research Questions of Section 1.4, and served as a guide for the workflow of the thesis work. As described in Chapter 6, and brought up again in this Chapter, every Objective was not completed during the thesis, given the drawn out implementation process between **O3** and **O5**. The second prototype was still developed according to **O6**, as seen in Section 7.4, but Objectives **O7** and **O8** remained unfinished. If they were finished, however, it would be possible that more detailed results would be acquired during the accompanying user tests, which in turn would mean that the Research question could be answered with more certainty.

Given the definitions of the milestones, however, considering the second prototype as the final implementation of the thesis, they were fully completed, but with the large omission of additional user tests.

8.2 Research questions

Several parts of the tool were designed and implemented with the Research questions of Section 1.4 in mind. This was also the case for the evaluation of the tool, covered in Chapter 5, where certain parts were included with the Research questions as a guide. Based on the results presented in Chapter 6 and discussed in Chapter 7, the following answers to the three Research questions have been found:

RQ1 What visualization methods are best suited to show a user where specific information can be found within a given text document or corpus?

Given the results on the close reading view, presented in Section 6.4.1, the highlighting and selection functionalities was a suitable approach to allow users to find specific information within given texts. However, the usability of this functionality was further improved through the use of additional visualization methods which provided extra data about that information. It can therefore be said that utilization of a close reading view only, still with the aforementioned functionalities, would be a worse approach compared to the cross-view-connection approach used in both prototypes. In addition to this, on the topic of different visualization methods, based on the initial encoding design in Section 3.3 it was highlighted that a vertical bar chart would be most suitable for representation of word frequency data, given the accompanying objectives. This choice was also strengthened by the quantitative result where the word frequency view was generally seen as usable, along with the qualitative feedback which described the view as easy to understand and a good feature.

Inclusion of less *traditional* visualization methods, in this case the word trends view, did not improve the usability of the tool, without proper information about its functionalities, as discussed in Section 7.3.3. On the other hand, given the cross-view highlighting of selected words, users could quickly find which segments of a text a selected word was significant to, thus showing that it was a suitable technique for users to locate specific information within documents.

The possibility to locate specific information within a given corpus was however less explored, compared to single document analysis, but a relevant method to this was decided in the design phase to be a node graph over similarities within the corpus. Similarity on its own was, however, considered to be to little information to allow the user to locate specific information within the corpus. Thus, as described in Section 4.7, the similarity overview was also implemented to provide information why a given document pair was similar. As discussed in Section 7.3.4, this was a suitable addition since the similarity overview was considered to be more usable according to the participants, and whose tasks was generally completed faster compared to the node view. Therefore, the node view was not suitable on its own as a method to locate specific information, but the additional information provided in the similarity overview did make up for this, provided that users used them conjunction with one another to find similar documents and the specific words which contributed to it.

RQ2 Which natural language processing methods are best suited for lightweight text analysis?

As described in Section 4.2, there were four text analysis methods implemented for the tool, two in regard to document specific analysis and two about corpus-specific analysis. In regard to the implemented document-specific methods, the *TF-IDF* method had on average a lower processing time compared to the *substrings* method, thus indicating that it would be better suited for lightweight text analysis from a performance perspective. The *TF-IDF* did also produce more usable visual representations in regard to the word repetition information which it could create, as presented in Section 4.2. Thus, in this context, the *TF-IDF* method is better suited for document-specific lightweight text analysis.

Among the corpus-specific analysis method, it was found that, in regard to performance that Jaccard similarity was slower to evaluate compared to cosine similarity. Additionally, cosine similarity was also better at creating language-based clusters within the node view, which according to the clustering objective mentioned in Section 3.2, was a strength in that method compared to Jaccard similarity. Thus, mostly given the points on performance and possibilities for differentiation between clusters, the cosine similarity method was better suited as a lightweight text analysis method for corpus-specific analysis.

All methods described here is still present in the second prototype of the tool however since they could produce visual representations.

RQ3 Should distant reading views be presented to the user initially alongside the close reading view, or are they more useful if provided on demand only?

Given the description found in the introduction of Chapter 6, some parts of the workflow were not completed in time for the user tests, which also extended to the additional layouts described in Section 4.10. These were designed with the third Research question in mind but were never implemented, meaning that no user tests were conducted with those layouts. However, based on the discussed feedback in Section 7.3.1 about the rather large size of the close reading view, it can be concluded that the alternative layouts found in Figures 4.13 and 4.14 contained wrong assumptions on the importance of a large close reading view. Additionally, the presented idea that user-defined scales of each view, discussed in the same Section, would be preferred, indicating that the tool should rely on this instead of the predefined view sizes provided by all layout examples found in Section 4.10. Lastly however, regarding what views should be visible to the user there was the general sentiment, presented in Section 6.4.1, that the layout as a whole was well-defined and followed a consistent structure. Related to this sentiment was the point of feedback, brought up in Section 6.4.1, that specified that the amount of data displayed in the layout of the first prototype was a preferred feature. It can therefore be said that, yes, the distant reading views should be presented alongside the close reading view, but with the addition that the close reading view should generally be smaller than the distant reading views and that user-defined view scaling would be welcomed.

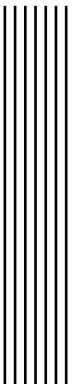
8.3 Future work

Comparison between the discussed improvements in Section 7.3 with the implemented features of the second prototype in Section 7.4 does provide some areas of future work for the *Lightweight TextVis* tool. For one, implementation of scalable views would further increase the usability in the tool, since each user could define what view they would like to have as their "main" view according to the demands of each user, as discussed in Section 7.3.1. Moving on, based on colour palette feedback of the similarity overview in Section 7.3.4, further connection between the node view and similarity overview could be established by colouring the grids based on the colours of the node pair. The current solution where the nodes are coloured through a linear interpolation between orange and blue would need to be changed however with this solution, since the contrast between the left and right grids in the similarity overview would be lost. Lastly on visual changes, tool tips for shortened words, a problem presented in Section 6.4.5, would preferably be included to provide users with additional information if desired.

Apart from visual improvements, another aspect to be improved in the tool is optimization to lessen processing times, a problem brought up in Section 6.4.1. It was attempted to avoid these problems with loading bars in the second prototype, but they never worked as expected and feedback on when the tool is processing files is thus not present. Given that the n-grams method never was implemented correctly, as described in Section 6.1, another improvement for the tool would be this additional *NLP* method. This would provide the users with additional methods to analyse their corpora, and maybe, based on the described

differences between the methods in Section 6.1.1, would allow them to find new information about said corpora.

With further work on the tool, however, further evaluation would also be needed. Since the second prototype currently is untested, it should be subjected to the same user tests as described in Chapter 5, and thus completing Objective **O7**. In addition to this, if faults are found during these tests, further development and additional tests with an accompanying evaluation phase, thus finally completing Objective **O8** as well.



Bibliography

[1] Kostiantyn Kucher and Andreas Kerren. "Text visualization techniques: Taxonomy, visual survey, and community insights". In: *2015 IEEE Pacific Visualization Symposium (PacificVis)* (2015), pp. 117–121. DOI: 10.1109/PACIFICVIS.2015.7156366.

[2] S. Jänicke, G. Franzini, M. F. Cheema, and G. Scheuermann. "Visual Text Analysis in Digital Humanities". In: *Computer Graphics Forum* 36.6 (2017), pp. 226–250. DOI: https://doi.org/10.1111/cgf.12873.

[3] Stéfan Sinclair and Geoffrey Rockwell. "Text Analysis and Visualization". In: *A New Companion to Digital Humanities*. Online tool: https://voyant-tools.org/. John Wiley & Sons, Ltd, 2015. Chap. 19, pp. 274–290. DOI: https://doi.org/10.1002/9781118680605.ch19.

[4] Kostiantyn Kucher, Carita Paradis, and Andreas Kerren. "DoSVis: Document Stance Visualization". In: *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - IVAPP*. INSTICC. SciTePress, 2018, pp. 168–175. ISBN: 978-989-758-289-9. DOI: 10.5220/0006539101680175.

[5] Alexis Deveria and Lennart Schoors. *JavaScript (initial core language support)*. Accessed: 2026-01-19. URL: https://caniuse.com/wf-javascript.

[6] K. R. Chowdhary. "Natural Language Processing". In: *Fundamentals of Artificial Intelligence*. Springer India, 2020, pp. 603–649. ISBN: 978-81-322-3972-7. DOI: 10.1007/978-81-322-3972-7_19.

[7] Nan-Chen Chen, Michael Brooks, Rafal Kocielnik, Sungsoo (Ray) Hong, Jeff Smith, Sanny Lin, Zening Qu, and Cecilia Aragon. "Lariat: A Visual Analytics Tool for Social Media Researchers to Explore Twitter Datasets". In: *Proceedings of the 50th Hawaii International Conference on System Sciences* (2017). DOI: 10.24251/HICSS.2017.228.

[8] Chengzhi Zhang, Huilin Wang, Yao Liu, Dan Wu, Y. P. Liao, and Bo Wang. "Automatic Keyword Extraction from Documents Using Conditional Random Fields". In: *Journal of Computer Information Systems* (2008).

[9] Jonathan D. Cohen. "Highlights: Language- and domain-independent automatic indexing terms for abstracting". In: *Journal of the American Society for Information Science* 46.3 (1995), pp. 162–174. DOI: 10.1002/(SICI)1097-4571(199504)46:3<162::AID-ASI2>3.0.CO;2-6.

[10] Annie Zaenen and Hans Uszkoreit. "Language Analysis and Understanding". In: *Survey of the State of the Art in Human Language Technology, Web Edition*. Cambridge University Press, 1997. Chap. 3. URL: https://www.dfki.de/~haus00/HLT-Survey.pdf.

[11] Shitanshu Jain, Sapan Kumar Jain, and Somil Vasal. "An Effective TF-IDF Model to Improve the Text Classification Performance". In: *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*. 2024. DOI: 10.1109/CSNT60213.2024.10545818.

[12] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. "Cosine similarity to determine similarity measure: Study case in online essay assessment". In: *2016 4th International Conference on Cyber and IT Service Management*. 2016. DOI: 10.1109/CITSM.2016.7577578.

[13] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. "Using of Jaccard Coefficient for Keywords Similarity". In: 2013.

[14] Paolo Nesi, Gianni Pantaleo, and Gianmarco Sanesi. "A Distributed Framework for NLP-Based Keyword and Keyphrase Extraction From Web Pages and Documents". In: *Distributed Multimedia Systems* (2015).

[15] Stefan Riezler and Michael Hagmann. "Validity, Reliability, and Significance: Empirical Methods for NLP and Data Science". In: Draft version. Springer International Publishing, 2022. Chap. 1. DOI: 10.1007/978-3-031-02183-1.

[16] Joel Coffman and Alfred C. Weaver. "A framework for evaluating database keyword search strategies". In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, 2010, pp. 729–738. DOI: 10.1145/1871437.1871531.

[17] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. "Novelty and diversity in information retrieval evaluation". In: Association for Computing Machinery, 2008, pp. 659–666. DOI: 10.1145/1390334.1390446.

[18] Robert Spence. "Information Visualization An Introduction Third Edition". In: Springer International Publishing, 2014. Chap. 1-4. DOI: 10.1007/978-3-319-07341-5.

[19] Niklas Elmqvist and Ji Soo Yi. "Patterns for visualization evaluation". In: *Information Visualization* 14.3 (2015), pp. 250–269. DOI: 10.1177/1473871613513228.

[20] Sheelagh Carpendale. "Evaluating Information Visualizations". In: *Information Visualization: Human-Centered Issues and Perspectives*. Ed. by Andreas Kerren, John T. Stasko, Jean-Daniel Fekete, and Chris North. Springer Berlin Heidelberg, 2008, pp. 19–45. DOI: 10.1007/978-3-540-70956-5_2.

[21] Google. *Books Ngram Viewer*. Accessed: 2026-01-23. URL: https://books.google.com/ngrams/.

[22] Marti A. Hearst. *Search User Interfaces*. Cambridge University Press, 2009. Chap. 11. DOI: 10.1017/CBO9781139644082.

[23] Helen C. Purchase. "Experimental Human–Computer Interaction - A Practical Guide with Visual Examples". In: Cambridge University Press, 2012. Chap. 1. DOI: 10.1017/CBO9780511844522.

[24] International Organization for Standardization. "ISO 9241-11:2018, Part 11: Usability: Definitions and concepts". In: *Ergonomics of human-system interaction* (2018).

[25] Erik Frøkjær, Morten Hertzum, and Kasper Hornbæk. "Measuring usability: are effectiveness, efficiency, and satisfaction really correlated?" In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2000. DOI: 10.1145/332040.332455.

[26] Thomas K. Landauer. "Research Methods in Human-Computer Interaction". In: *Handbook of Human-Computer Interaction 1st Edition*. 1988. Chap. 42.

[27] Tamara Munzner. "A Nested Model for Visualization Design and Validation". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 921–928. DOI: 10.1109/TVCG.2009.111.

[28] S. Wehrend and C. Lewis. "A problem-oriented classification of visualization techniques". In: *Proceedings of the First IEEE Conference on Visualization: Visualization '90*. 1990, pp. 139–143. DOI: 10.1109/VISUAL.1990.146375.

[29] Eliane R. A. Valiati, Marcelo S. Pimenta, and Carla M. D. S. Freitas. "A taxonomy of tasks for guiding the evaluation of multidimensional visualizations". In: *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. Association for Computing Machinery, 2006, pp. 1–6. DOI: 10.1145/1168149.1168169.

[30] Ben Shneiderman. "The eyes have it: a task by data type taxonomy for information visualizations". In: *Proceedings 1996 IEEE Symposium on Visual Languages*. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307.

[31] Jock Mackinlay. "Automating the design of graphical presentations of relational information". In: *ACM Trans. Graph.* 5.2 (1986), pp. 110–141. DOI: 10.1145/22949.22950.

[32] Catherine Plaisant. "The challenge of information visualization evaluation". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. Association for Computing Machinery, 2004, pp. 109–116. DOI: 10.1145/989863.989880.

[33] Adam Perer and Ben Shneiderman. "Integrating Statistics and Visualization for Exploratory Power: From Long-Term Case Studies to Design Guidelines". In: *IEEE Computer Graphics and Applications* 29.3 (2009), pp. 39–51. DOI: 10.1109/MCG.2009.44.

[34] Tshephisho Joseph Sefara, Mahlatse Mbooi, Katlego Mashile, Thompho Rambuda, and Mapitsi Rangata. "A Toolkit for Text Extraction and Analysis for Natural Language Processing Tasks". In: *2022 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*. 2022, pp. 1–6. DOI: 10.1109/icABCD54961.2022.9856269.

[35] brython-dev. *Brython*. Accessed: 2026-01-29. URL: https://brython.info/index.html.

[36] Thord Lindé. *www.bellman.net*. Swedsish website, Accessed: 2026-03-11. URL: https://www.bellman.net/texter/index.html.

[37] Litteraturbanken. *Litteraturbanken*. Swedsish website, Accessed: 2026-03-11. URL: https://litteraturbanken.se/bibliotek.

[38] Project Runeberg volunteers. *Project Runeberg*. Accessed: 2026-03-11. URL: https://runeberg.org/.

[39] Finansinspektionen. *Reports*. Accessed: 2026-03-11. URL: https://www.fi.se/en/published/reports/.

[40] Project Gutenberg volunteers. *Project Gutenberg*. Accessed: 2026-03-11. URL: https://www.gutenberg.org/.

[41] Office for National Statistics. *About us*. Accessed: 2026-03-11. URL: https://www.ons.gov.uk/aboutus.

[42] Uppsala University. *DiVA portal*. Accessed: 2026-03-11. URL: https://www.info.diva-portal.org/w/diva/en/about-diva.

[43] MDN. *Unicode character class escape*. Accessed: 2026-03-12. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Regular_expressions/Unicode_character_class_escape.

[44] Regular expressions 101. *Quick Reference, Common Tokens*. Accessed: 2026-03-12. URL: https://regex101.com/.

[45] Steffen Koch, Markus John, Michael Wörner, Andreas Müller, and Thomas Ertl. "VarifocalReader — In-Depth Visual Analysis of Large Text Documents". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014). DOI: 10.1109/TVCG.2014.2346677.

[46] Kostiantyn Kucher, Teri Schamp-Bjerede, Andreas Kerren, Carita Paradis, and Magnus Sahlgren. "Visual analysis of online social media to open up the investigation of stance phenomena". In: *Information Visualization* 15.2 (2016). DOI: 10.1177/1473871615575079.

[47] Elijah Meeks and Anne-Marie Dufour. *D3.js in Action, Third Edition*. Manning Publications, 2024. ISBN: 9781633439177.

[48] Cynthia Brewer and Mark Harrowe. *COLORBREWER 2.0*. Accesed: 2026-02-03. URL: https://colorbrewer2.org/.

[49] Kostiantyn Kucher and Andreas Kerren. "Text visualization techniques: Taxonomy, visual survey, and community insights". In: *2015 IEEE Pacific Visualization Symposium (PacificVis)*. 2015. DOI: 10.1109/PACIFICVIS.2015.7156366.

[50] Daniel A. Keim and Daniela Oelke. "Literature Fingerprinting: A New Method for Visual Literary Analysis". In: *2007 IEEE Symposium on Visual Analytics Science and Technology*. 2007. DOI: 10.1109/VAST.2007.4389004.

[51] fergie and Espen Klem. *stopword*. Accessed: 2026-04-29, Version 3.1.5. 2025. URL: https://www.npmjs.com/package/stopword.

[52] SIL Global. *ISO 639 Code Tables*. Accessed: 2026-04-29. URL: https://iso639-3.sil.org/code_tables/639/data/all.

[53] Titus Wormer. *iso-639-3*. Accessed: 2026-04-29, Version 3.0.5. 2022. URL: https://www.npmjs.com/package/iso-639-3.

[54] Vic Costello. *Multimedia Foundations: Core Concepts for Digital Design (3rd ed.)* Focal Press, 2023. Chap. 5. DOI: 10.4324/9780429422669.

[55] Saugat Pandey and Alvitta Ottley. "Mini-VLAT: A Short and Effective Measure of Visualization Literacy". In: *Computer Graphics Forum* 42.3 (2023), pp. 1–11. DOI: https://doi.org/10.1111/cgf.14809.

[56] Keith Andrews. "Evaluating information visualisations". In: *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. Association for Computing Machinery, 2006. DOI: 10.1145/1168149.1168151.

[57] Tamara Munzner. *Visualization Analysis and Design*. CRC Press, 2015. DOI: 10.1201/b17511.

[58] Malcolm Koo and Shih-Wei Yang. "Likert-Type Scale". In: *Encyclopedia* 5 (2025). DOI: 10.3390/encyclopedia5010018.

[59] Jonas Blattgerste, Jan Behrends, and Thies Pfeiffer. "A Web-Based Analysis Toolkit for the System Usability Scale". In: PETRA '22. Association for Computing Machinery, 2022. DOI: 10.1145/3529190.3529216.

[60] Cambridge University Press & Assessment. *International language standards*. Accessed: 2026-05-19. 2026. URL: https://www.cambridgeenglish.org/exams-and-tests/cefr/.

[61] Jean-Daniel Fekete and Juliana Freire. "Exploring Reproducibility in Visualization". In: *IEEE Computer Graphics and Applications* 40.5 (2020), pp. 108–119. DOI: 10.1109/MCG.2020.3006412.

[62] Council of European Union. *Council regulation (EU) no 2016/679*. 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng.

[63] Plamen Milev. "THE ROLE OF DATA VISUALIZATION IN ENHANCING TEXTUAL ANALYSIS". In: *Trakia Journal of Sciences* 21 (2023), pp. 248–253. DOI: 10.15547/tjs.2023.s.01.042.

A Appendix

Testing plan for the Lightweight TextVis Tool
Equipment: Laptop with a screen resolution of 1920x1080, tool running locally, a computer mouse (partly given slightly faulty trackpad and to allow for more precise inputs).

Interviewer: *"Welcome to the test for the Lightweight TextVis tool. The motivation behind this tool is to provide the user with both close and distant reading methods for singular documents within a provided corpus. This is meant to be provided in a generalized setting, meaning that the tool will make as few assumptions as possible on the provided data. Are you ready to partake in this user test* **[YES/NO]***?"*

1. Introduction of the general ideas behind the tool.
    - Andrews highlights the weakness *Familiarity of test users with traditional interfaces* during user tests. Attempt to avoid this by explaining what the tool aims to achieve and in broad strokes what it can be used for [56].
    - Is also included to provide the user with required information before the test. This choice is based on Carpendale's advice to the interviewer to talk less during the test [20].

I: *"Based on proposed target user bases for the tool, I would first like to ask you about your age* **[Let the test participant answer]**. *Your highest level of education, and if that has finished* **[Let the test participant answer]**, *and finally, your English language proficiency, since the instructions present in the tool are written in English* **[Let the test participant answer]**."

2. Question about test participants age, and highest level of education, and language proficiency (relevant to written help in the tool).
    - Included given the described target user base in the design of the tool. Language proficiency is also questioned about since the entire tool is written in English, a test participant which doesn't speak the language fluently might complete tasks slower compared to a native speaker.

I: *"Before you are provided with the tool and task for that tool, I'd like you to complete the Mini-VLAT, this purely aims to answer how familiar you are with general data visualization methods, and will take 5 minutes at a maximum* **[Test participant takes the test]**.*"*

3. Completion of the Mini-VLAT to evaluate the test participants' visualization literacy.
    - Included as a measure on how familiar a test participant is with general visualization methods, results of the user test can be skewed on this previous familiarity [20].
    - The measurement does go the other way as well however. If a test participant scores low on the Mini-VLAT, they might have difficulties understanding the tool [55].

I: *"Good, I'll now provide you with the tool* **[Change tabs of the browser to preloaded tool]**.*"*

4. Load test data to be used (Done in advance given the current processing time).
    - Ensure to load documents in English, Swedish, and possibly other languages. Will populate the node graph and corpus overview.
    - There will still be significant slowdowns during the test, but by loading the documents in advance, a larger problem can be avoided.
    - Generally across the test, processing times will be used to evaluate if the user prefers loading bars or some feedback when the tool stays "still".

I: *"The first task of this test will simply handle navigation in the tool, I'd like you to explore the available views and windows of the tool. Tell me when you identify a separate view or window available, if needed I will also help you to find any that have been missed* **[The test participant is allowed to freely navigate in the tool]**.*"*

5. General navigation test, can the user find all views on their own accord. Note down time taken, along with any assistance needed to find specific views. They are tasked with exploring the tool.
    - As expressed in the description, this test is quantitative given that the test participant is tasked with finding the different views in the tool and their time will be noted.
    - There are five views accessible in the tool, these being: Close Reading view, Word Frequency view, Word Trends view, Node Graph view, Corpus overview, and Similarity overview. The four main views should be found immediately, since they are always displayed. The Corpus and Similarity overviews are however only accessible through clicking, this task thus aims to find if the user understands how to access these views along with if these are accessible from suitable locations.
    - Based on Carpendale's Quantitative Methodology the hypothesis is that the separation between different views, along with them having different titles, is enough to tell them that they display different data based on the same input [20].

I: *"Good, so now that you have familiarized yourself with the tool, I'd like to know how you believe the different views you see are connected to one another? For example, what views display document specific data and what views display corpus specific data?* **[Let the test participant answer]**.*"*

6. After exploration of the tool, question about how the different views are connected, can the user identify this with limited interaction?

    - Limited interaction in this case refers to things like hovering or just observing the different views.Can the user identify that the close reading, word frequency, and word trends views display the evaluated data for the same document. Is it possible to understand that the corpus overview and node graph shows the same corpus as well?
    - Limiting the expected number of interactions from the user is meant to serve as a form of Control of Independent Variables given that this task only aims to study whether or not the test participant can understand the connections, not the usability of said connections [20].

I: *"For your next task I'd like to find and identify the three different ways you can use to change what document is displayed, it's okay that you change the document with an identified method to confirm how it works* **[Let the test participant explore again, note down if they change document, since this can effect the time taken]**."

7. Corpus navigation, task the user with finding the three different ways to change the currently displayed document.

    - The first interaction task (clicking and scrolling) is also meant to control an independent variable of how quickly the user can understand and navigate through the provided corpus [20].
    - The three implemented navigation methods are through the Document dropdown in the settings bar, clicking on the Document cards in the corpus overview, and clicking on the Document nodes in the node graph.
    - All of the elements that can be used for corpus navigation has been numbered according to their index in the corpus array, meant to highlight that these items are connected, and have similar interaction techniques available to them.

I: *"Now that you've found these methods, I'd like you to navigate to Henrik Ibsen's "A dolls house" with your preferred method* **[Let the test participant navigate to "A dolls house"]**."

8. Task the user to navigate to a specific document in the corpus with their desired method.

    - Since the test participant hopefully has found all navigation options at this point, this task is not timed.
    - Since the test participant hopefully has found all navigation options at this point, this task is not timed.
    - Important to note here is also if the test participant chose the last method they viewed. If the trends point towards this being the case, there is the risk that their choice largely is dependent on a familiarity bias [20].

I: "The two following tasks involve finding specific information within this document. To begin I'd like you to find the fifth occurrence of the word "Money". I'll provide you with the hint that the word has a total of 48 occurrences in this text **[Let the test participant find the specific word, stop time when it's possible to see that it has been selected]**."

9. Document navigation, in the chosen document, the user is tasked with finding a specific word/token, additionally, making them find a specific occurrence of said word in the document (tests if the occurrence navigation in the frequency view is intuitive). A hint about the number of occurrences of the given word will also be given, in order to test if the occurrence filtering works as expected.
    - This task is timed in order to find quantitative data on how quick the user can find specific words in a given document.
    - Since the test participant hopefully has found all navigation options at this point, this task is not timed.
    - The hypothesis is that the frequency view is usable for the objective of locating in the close reading view. Independent variables are controlled by tasking all test participants to find the same word and occurrence within the same document [20].

I: *"With the word found I'd like you to answer how many segments of the text this word is considered significant in. Additionally, in what segment is the word considered to be most significant among those it can be found in* **[Let the test participant search the word trends view and note down if they are correct or not, is significant to three segments, occurance in second segment is most significant]***?"*

10. Question the user on the number of segments this word is significant in, along with tasking them to find in which segment it's most significant.
    - This ties to the previous task in control of independent variables, it is absolutely necessary to task them with finding a word which has several occurrences in the word trends view, otherwise the task does not work.
    - Here they will be timed on how quick they can answer the given questions correctly, along with noting down the failure rate if applicable.
    - Aims to find if the colour scheme used in the word trends view has enough contrast (the test participant can make out the different items) and if the repetition of colours for identical items relates the information that said item is significant in several segments [57].

I: "Moving on to the next set of tasks, I'd like you to find which two documents in the corpus are considered to be most similar. If you find the view to be too cluttered it is possible to filter the similarity in the settings **[Let the test participant find the largest similarity in the corpus (found between the Capital Stocks And Fixed Capital Consumption reports from the years 2023 and 2024), note down if they use the filtering method, since this does increase the time taken]***."*

11. Task the user to find which documents are most similar in the corpus, if the web of nodes is too dense, hint that it's possible to filter the node graph.
    - Also a time dependent task which aims to find if the edge sizes are distinguishable and if the test participant can understand that their size correlates to the similarity score [29].
    - The hint will be given to all test participants, and it will be noted down if the participant uses the filtering method, and can indicate that it is a useful feature.

I: *"Good, I'd now like to ask you the top five words which contribute to this similarity* **[Let the test participant view and use the similarity overview. Correct list: Capital, Net, Of, The, And]***?"*

12. Question about which top five words/tokens contributes the most to this similarity, based on the provided scores.

    - Timed task to evaluate the colour choices in the similarity overview since this information can be found in the yellow squares of the document grids.
    - It is however important to note that this information can also be acquired through the similarity table, thus asking the participant which method they used to find the answer.

I: *"That concludes the tasks focusing on specific views. I'd now like you to return to the main view of the tool (the starting screen) and I'd like to know what you believe the different views display? For example what data each view presents and what they aim to provide to the user* **[Let the test participant answer and note down said answers]**.*"*

13. General questions about what the different views represent, according to the test participant themself.

    - Given that they now have used the main features of the tool, what information do they think is displayed in each view, mostly for collection of qualitative data. Their answers will however also be evaluated to acquire quantitative data to measure how many of these answers are correct based on the intended use of the views.
    - Answers about the node graph, corpus overview and similarity overview will also be evaluated in a similar manner.

I: *"And that concludes the test, is there any additional feedback you'd like to provide? Possibly thoughts about the layout of the views and windows, or the colour choices within the tool* **[Let the test participant answer]***?"*

14. Ask about additional feedback.

    - If the tasks failed to highlight specific thoughts from the test participant, they can give that feedback here at the end. This is meant to be purely qualitative data collection, and serves as general points of improvement for the tool.

I: *"Finally, I'd like you to complete this form in order to get formulated answers about your experience with the tool now that you've used it* **[Provide the user with a Google forms form, wait for them to answer]**. *Thank you for your participation and time."*

15. Ask the user to complete a *SUS*-test at the end of the test, along with asking them for opinions about the tool which might not have been answered yet.

    - Done to measure the test participants perceived Effectiveness, Efficiency, and Satisfaction with the tool during the test [25].
    - Will be constructed as a likert scale to give quantitative data about the different views and the tool in general as perceived by the user.

B Appendix

- **Carl Michael Bellman**
  - *No:1 FREDMANS EPISTEL*, words: 270
    https://www.bellman.net/texter/epistel.php?nr=1
  - *No:2 FREDMANS EPISTEL*, words: 216
    https://www.bellman.net/texter/epistel.php?nr=2
  - *No:3 FREDMANS EPISTEL*, words: 219
    https://www.bellman.net/texter/epistel.php?nr=3
  - *No:4 FREDMANS EPISTEL*, words: 161
    https://www.bellman.net/texter/epistel.php?nr=4
  - *No:5 FREDMANS EPISTEL*, words: 168
    https://www.bellman.net/texter/epistel.php?nr=5
  - *No:6 FREDMANS EPISTEL*, words: 134
    https://www.bellman.net/texter/epistel.php?nr=6
  - *No:7 FREDMANS EPISTE*L, words: 258
    https://www.bellman.net/texter/epistel.php?nr=7
  - *No:8 FREDMANS EPISTEL*, words: 205
    https://www.bellman.net/texter/epistel.php?nr=8
  - *No:9 FREDMANS EPISTEL*, words: 354
    https://www.bellman.net/texter/epistel.php?nr=9
  - *No:10 FREDMANS EPISTEL*, words: 192
    https://www.bellman.net/texter/epistel.php?nr=10
- **Anton Chekhov**
  - *The Bet, and other stories*, words: 59358
    https://www.gutenberg.org/ebooks/55283
  - *The sea-gull*, words: 22337
    https://www.gutenberg.org/ebooks/1754

- **Henrik Ibsen**
  - *A Doll's House : a play* , words: 30327
    https://www.gutenberg.org/ebooks/2542
- **Franz Kafka**
  - *Metamorphosis*, words: 25500
    https://www.gutenberg.org/ebooks/5200
  - *The Trial*, words: 89454
    https://www.gutenberg.org/ebooks/7849
- **Office for National Statistics**
  - *Capital stocks and fixed capital consumption, UK: 2023*, words: 1729
    https://www.ons.gov.uk/economy/nationalaccounts/uksectoraccounts/bulletins/capitalstocksconsumptionoffixedcapital/2023
  - *Capital stocks and fixed capital consumption, UK: 2024*, words: 2160
    https://www.ons.gov.uk/economy/nationalaccounts/uksectoraccounts/bulletins/capitalstocksconsumptionoffixedcapital/2024
  - *Capital stocks and fixed capital consumption, UK: 2025*, words: 2571
    https://www.ons.gov.uk/economy/nationalaccounts/uksectoraccounts/bulletins/capitalstocksconsumptionoffixedcapital/2025

C Appendix

Lightweight TextVis User Survey
Survey for the Lightweight TextVis tool to measure usability

Questions relating to the general layout of the tool and menu navigation

Effectiveness: The placement of different views and navigation buttons allowed for accuracy in finding and comparing data.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Efficiency: The placement of different views and navigation buttons allowed for data searching and comparison in a suitable time frame.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Satisfaction: The placement of different views and navigation buttons provided a positive experience for data searching and comparison.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Additional thoughts about the general layout of the tool along with the menu navigation.

Questions relating to the close reading view

Effectiveness: The coloured highlights and segment breaks was a suitable method to accurately find specific words.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Efficiency: Navigation of a text (scrolling or clicking on words) within the close reading view provided a way to find specific words in a suitable time frame.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Satisfaction: The text formatting and colour choices within the close reading view provided a positive experience for reading and searching for specific words.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Additional thoughts about the close reading view.

Questions relating to the word frequency view

Effectiveness: The provided methods (scrolling and filtering) and structure of the chart meant that the occurrence of specific words could accurately be found.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Efficiency: The frequency of a desired word could be found in a suitable time frame with the help of provided methods and the structure of the chart.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Satisfaction: The experience of searching for a specific word and its frequency in the document was positive.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Additional thoughts about the word frequency view.

Questions relating to the word trends view

Effectiveness: The colour choices in the stacked bar chart meant that the significance of certain words could be accurately deciphered.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Efficiency: The highlighting features and colour repetition throughout the stacked bar chart meant that segments which a desired word was significant to could be found in a suitable time frame.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Satisfaction: The experience of searching for segments which a desired word was significant to was positive.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Additional thoughts about the word trends view.

Question relating to the connection between document specific views

Effectiveness: The connected highlighting features and shared colour schemes between the close reading, frequency, and trends views meant that desired words could be accurately located within the displayed document.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Efficiency: Locating desired words within the displayed document could be done in a suitable time frame with the help of the tools in the different views.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Satisfaction: Utilizing the connected features between the views to locate desired words in the displayed document was a positive experience.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Additional thoughts about the connection between the document specific views.

Questions relating to the node graph view

Effectiveness: The colours of nodes and connections between them allowed for accurate comparison of document similarities.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Efficiency: The colour of nodes and connections between them allowed for similarity scores to be found in a suitable time frame.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Satisfaction: The provided navigation features, colour choices, and highlighting methods contributed to a positive experience when using the node graph view.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Additional thoughts about the node graph view.

Questions relating to the similarity overview

Effectiveness: The structure of the tables and grid overviews provided a way to accurately find the reason why two documents were considered similar.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Efficiency: Contributing factors to the document similarity could be found within a suitable time frame.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Satisfaction: The experience of utilizing tables and grid overviews to understand why documents were considered to be similar was positive.

Strongly Disagree	1	2	3	4	5	Strongly Agree
	◯	◯	◯	◯	◯	

Additional thoughts about the similarity overview.

Questions relating to corpus navigation

Effectiveness: The provided methods to change the displayed document meant that desired documents could accurately be selected.

Strongly	1	2	3	4	5	Strongly
Disagree	◯	◯	◯	◯	◯	Agree

Efficiency: Changing the displayed document through corpus navigation could be achieved in a suitable time frame.

Strongly	1	2	3	4	5	Strongly
Disagree	◯	◯	◯	◯	◯	Agree

Satisfaction: The experience of selecting a document and changing the displayed document was positive.

Strongly	1	2	3	4	5	Strongly
Disagree	◯	◯	◯	◯	◯	Agree

Additional thoughts about the corpus navigation.