Master Thesis Project

Interactive Multiscale Visualization of Large, Multi-dimensional Datasets

Author: Kay KÜHNE
Supervisor 1: Prof. Dr. Andreas KERREN
Supervisor 2: Dr. Rafael M. MARTINS
Examiner: Prof. Dr. Welf LÖWE
Semester: VT2018
Course code: 4DV50E
Subject: Computer Science

Abstract

This thesis project set out to find and implement a comfortable way to explore vast, multi-dimensional datasets using interactive multiscale visualizations to combat the ever-growing information overload that the digitized world is generating. Starting at the realization that even for people not working in the fields of information visualization and data science the size of interesting datasets often outgrows the capabilities of standard spreadsheet applications such as Microsoft Excel. This project established requirements for a system to overcome this problem. In this thesis report, we describe existing solutions, related work, and in the end designs and implementation of a working tool for initial data exploration that utilizes novel multiscale visualizations to make complex coherences comprehensible and has proven successful in a practical evaluation with two case studies.

Keywords: Data Exploration, Visual Analytics, Multiscale Visualization, Focus+Context, Overview+Detail

Acknowledgements

I would like to thank my two supervisors, Prof. Dr. Andreas Kerren and Dr. Rafael M. Martins, for providing this fascinating topic, and their help and guidance along the whole project. I also would like to thank all attendees of my expert interview for sharing some of their precious time with me and helping me in drawing a conclusion for this project. Last but not least I want to thank my two opponents for giving me valuable feedback to improve my thesis report and my examiner for taking the time to evaluate it.

Contents

List of Figures

List of Tables

List of Abbreviations

API	**A**pplication **P**rogramming **I**nterface
BI	**B**usiness **I**ntelligence
HASC	**H**ierarchical **A**dministrative **S**ubdivision **C**odes
JSON	**J**avaScript **O**bject **N**otation
NTS	**N**ordic **T**weet **S**tream
RQ	**R**esearch **Q**uestion
TTI	**T**ime-**T**o-**I**nteractive
UEQ	**U**ser **E**xperience **Q**uestionnaire
URL	**U**niform **R**esource **L**ocator
UX	**U**ser e**X**perience
VA	**V**isual **A**nalytics

1. Introduction

The overall goal of this thesis project was to find a comfortable way to explore vast, multi-dimensional datasets. The target was not to provide a tool exclusively for data scientists, but a system that can also be used by people unfamiliar with information visualization and data exploration to conduct initial data exploration. To make the complex coherences in the datasets comprehensible, the system was required to apply novel multiscale visualization techniques and a concept to effortlessly filter through different dimensions of a dataset.

The following sections give an introduction to the background of this project, and explain the motivation behind the project in more detail. The initial problem is stated in depth and formulated into a research question. The approach to solving this problem and the expected results are laid out together with the target groups and the overall structure of this report.

1.1. Background

To understand the targeted problem, one needs to understand the field of **visualization**. It is defined as "the activity of forming a mental model of something" by R. Spence [1]. The field is split into three main areas: (i) *Information Visualization*, (ii) *Scientific Visualization*, and (iii) *Visual Analytics*.

1.1.1. Visual Analytics

Information Visualization is the visualization of abstract data to reinforce human cognition, or "the design of visual data representations and interaction techniques that support human activities where the spatial layout of the visual representation is not a direct mapping of spatial relationships in the data."[1]

Scientific Visualization is mainly about the "visualization of three-dimensional phenomena" [2]. As of its nature, it is considered a subset of computer graphics and has, for example, a large impact in medical research [3].

Visual Analytics (VA) is the science of coupling interactive visual representations with their underlying analytical processes [4]. The target is to provide technologies to understand not only the information itself, but also the underlying process that derived it from the raw data and therefore, to combine the strengths of human cognition and electronic data processing for the most effective results. This is crucial as the users do not have to trust the output of a machine blindly; VA enables them to understand the reasoning behind information and makes it easier to base heavy weighting real-life decisions on it. To achieve this, VA relies on highly interactive visual interfaces which pose in itself an important scientific challenge [4, 5]. Due to its interdisciplinary nature, VA also requires a scientific background in "statistics, mathematics, knowledge representation, management and discovery technologies, cognitive and perceptual sciences, decision sciences, and

[1]IEEE, IEEE InfoVis 2017 – Topics and Paper Types, 2017. [Online]. Available: http://ieeevis.org/year/2017/info/call-participation/infovis-paper-types. [Accessed: 28-Oct-2017]

more."[2] The whole area is a rather new one [1] and has some overlap with the other mentioned areas of visualization. In conclusion, VA is turning information overload into an opportunity and this is precisely the main goal of this project.

1.1.2. Multiscale Visualizations

The centerpiece of the frontend is a collection of interconnected and, when appropriate, multiscale visualizations. Multiscale visualization is a relatively new research topic, and the term is not unambiguously defined yet. Related work uses the term mostly for three different classes of techniques (and often applies more than one of those to the same visualization): *multiresolution*, *literal multiscale*, and *non-linear scale*. In general, multiscale visualizations are beneficial for the user experience, because they enable users to get an overview and, at the same time, *gradually* focus their view.

The **multiresolution** approach changes resolution inside a graphic [6]. This requires hierarchical data to group together in order to lower the resolution, allowing a focused area to be shown in finer detail without extracting it from its surrounding area. For example, a stacked area chart would change its granularity for a focused segment and show its subgroups instead of the overlying category, transitioning back into the category once the segment selection is over.

A **literal multiscale** visualization is more common and consists of applying the multiresolution approach but utilizing multiple graphics. For instance, for a stacked area chart on a time series scale this would mean having an overview graphic, visualizing the whole timeframe, and a second graphic visualizing a selected subset of this timeframe in more detail. A less complex example would be the displaying of different granularities for the same selection [7]. This technique is also called a multiresolution visualization with an *Overview+Detail* approach in related literature [8].

More seldom referenced as multiscale visualization is the approach of using a **non-linear scale**. In this approach, the scale changes inside the visualization to display a segment in more detail by providing it more physical space. One example of this is a time series graph displaying the means of some measured value for all months per year for the previous years, and then changing to a direct monthly aggregation for the current year, avoiding to break the visual flow of the chart by transforming the scale. One way to achieve this effect dynamically upon user interaction is the so-called *Focus+Context* approach [9].

1.1.3. Spatiotemporal Visual Analytics

Spatiotemporal visual analytics is a subset of VA that focuses on the visual analysis of spatiotemporal data [10], i.e., time-based data with a location dimension. For example, weather data or movement tracking qualify as spatiotemporal data. Spatiotemporal data is an interesting use case for multiscale visualization as the location offers a flexible and (for the user) obvious dimension to group data into different resolutions.

1.2. Motivation

With the rise of the Internet and the digitization of ever more areas, the amount of collected data is enormous and continually expanding [11]. While the storage capacities kept

[2]IEEE, VAST Papers, 2016. [Online]. Available: http://ieeevis.org/year/2016/info/call- participation/vast-papers. [Accessed: 28-Oct-2017]

growing with the data generation, the ability to use it could not keep up, leaving us with vast amounts of raw data with no value on its own [4]. The growing gap between the stored data and the ability to analyze it is broadly known under the term "information overload" [12]. As companies discover the value of their aggregated datasets, the economic field around data science is growing massively. However, there is a high hurdle for initial data exploration for people who are not proficient in this field. As soon as datasets outgrow Microsoft Excel's capacity, they also outgrow the capabilities of most users. Therefore, from a practical standpoint, there is a significant need for appropriate visualization and exploration tools by non-data-science users.

From a scientific standpoint, the proposed visualizations and the interconnected filtering between the different visualizations and dimensions provide novelty as they build upon current research work and drive the scientific field further. As for the field of multiscale visualizations, recent publications on the subject, such as an interactive multiscale visualization for streamgraphs by Cuenca et al. [13], show its novelty. As for the system as a whole, it will provide a stepping stone for further projects that involve large datasets, and is an integral part to support the visualizations of this project.

The state of the art in the industry regarding data exploration and visualization expanding over basic Microsoft Excel usage is that companies who have the necessary funding either keep in-house data-science employees or have custom-tailored solutions developed for their specific needs. There are multiple companies who specialize in providing such development and initial exploration to solvent customers. An example of that would be anacision[3] and EXXETA[4]. Other than that, there are highly specific solutions to problems that are the same across an industry. A good example here would be the *Business Intelligence* (BI) market which utilizes standardized data from so-called *Data Warehouses*. Data Warehouses store data that has been standardized by an underlying Integration Layer to provide its BI software with a solid and already enriched dataset. For the end user, the possibly best-known tool exceeding the abilities of Microsoft Excel is Tableau[5]. Tableau is an award-winning data analytics software with a focus on BI. Especially interesting for end users is the ability to import data quickly from sources like spreadsheets or relational databases.

The current state of research regarding multiscale visualization shows some promising results in the last years. In the mentioned paper from Cuenca et al. [13], the authors presented a solution to multiscale visualization for streamgraphs (advanced stacked time series) which uses multiple resolutions to split up the time series into subseries for a limited time series [13]. Interestingly, the visualization utilized all three techniques labeled under the term multiscale visualization, but only explicitly mentioned *multiresolution* in the associated poster [6]. Further related work is listed and analyzed in Chapter 2.

1.3. Problem Statement

The frontend is supposed to display various interconnected visualizations that adapt to the dataset, and—if fitting for the visualization—present the data in a multiscale way. Multiscale visualizations offer the opportunity to enhance the experience for the user significantly but are harder to develop as they are more complicated than their single-scale versions. They not only hold more complexity to visualize but also depend on specific queries and data aggregation in the backend. There already exist specific solutions for

[3]https://www.anacision.de. [Accessed: 23-Apr-2018]
[4]https://www.exxeta.com. [Accessed: 23-Apr-2018]
[5]https://www.tableau.com. [Accessed: 23-Apr-2018]

particular types of visualizations, but it is not clear if it is possible to adapt these to integrate seamlessly into an environment where each visual component influences the data displayed on the others.

Before the frontend is able to display even the most basic information, there needs to be a data-providing backend in place. There are three requirements concerning the backend: (i) the ability to adapt to different datasets, (ii) the ability to handle extensive and complex datasets, and (iii) to provide all that with a performance that does not break the focus of the user upon interactions.

To be able to adapt to different datasets, the backend has to be generic enough to deal with different problems and provide a protocol to communicate with the frontend in a manner that works with a wide range of different data types. With this ability, the backend as a whole will qualify as a framework for further projects.

The whole system needs to be able to handle extensive and complex datasets with a performance that does not interrupt the user. However, not only the large datasets themselves present a challenge; with the large sets and the high complexity, there is also a high possible variety of different filters. As it is desirable to provide most of this flexibility to the user, the system needs to be able to cope with a significant workload. Otherwise, the user would experience slow response time upon interacting with the frontend which directly leads to context disruptions that make it harder for the user to understand the process and the visualized information.

The user is supposed to understand and be part of the information finding process, but not to feel the workload the system has to handle. Therefore, the efficiency of the backend has a direct influence on the effectiveness of the whole system.

From these problem statements, we can derive the following central research question:

RQ *How can we explore large, multi-dimensional datasets using interactive multi-scale visualization both efficiently and effectively?*

To clearly understand the question, we have to define the terms *efficiency* and *effectiveness* for this context. As a guideline we can look at the definition for both terms in the ISO standard 9241 [14].

Effectiveness can be defined as the act of providing a truly insightful experience to the user through interactive visualization. This means enabling the user to explore the dataset on a level not reasonably achievable before.

Efficiency can be defined as the act of providing the needed performance to adapt to new filters and react to interactions. The lesser time the tool needs to provide the user with the requested view, the higher its efficiency. This performance definition can be quantifiably measured not only objectively with reaction times, but also as a direct influence on the effectiveness of the system as an unefficient system causes a unpleasant experience for the user.

1.4. Solution Approach

To find a solution to the stated problems and answer the research question, the project started based on the findings from research into related work and a collection of requirements, coupled with two proposed case studies. A system was designed following the requirements, implemented, and used to realize the case study. An experiment, a survey and expert interviews with think-aloud sessions were conducted to secure verification and validation of the system. As the scope of this project differs from the related projects, a full comparison was not possible, but concluding this report the differences in scope and resulting divergence were explained in detail.

1.5. Contributions and Target Groups

The main novelty and contribution of this project is the resulting system to explore vast, multi-dimensional datasets using interactive multiscale visualization. The system can be used for further projects and the technology and technique trade-off considerations and evaluation in this report can help to improve following versions and other tools. The system was designed to be quickly adaptable to new data sources and viewing angles, making it predestined for further usage. Possible target groups for the system are organizations interested in a more comfortable exploration of large and complex datasets without dedicated data scientists. Such organizations can span from research institutes to medium-sized companies. Due to the nature of the software structure and the encapsulated components, these can be plugged into different systems without any code changes to the component. This makes the project interesting for developers and software engineers already working with VA or planning to include VA in a project.

The multiscale visualizations in the frontend also provide scientific novelty. They were built to improve upon recent publications in this field introduced in Chapter 2 *Background*. The combined concept, together with the implementation and the integration into a whole framework, differentiates this work from previous publications which stand more as a proof-of-concept. The evaluation provides insight into the human acceptance of the concepts and offers suggestions for improvements and new concepts. This part is more targeted at researchers in the field of VA, but also interesting for developers and software engineers looking to include more complex visualizations into an already existing system.

1.6. Report Structure

The remainder of this report is divided into six chapters. First, related work is listed and analyzed in Chapter 2 *Background*. Afterwards, the utilized research methods are explained, and possible reliability, validity, and ethical considerations are discussed in Chapter 3 *Methodology*. In Chapter 4 *Conception*, the requirements of the system, the system design and the concept of the design behind the case studies are layed out. In Chapter 5 *Implementation*, the implementation of the system and the case studies are described and illustrated. The evaluation of the system is described in Chapter 6 *Evaluation* and conclusions are discussed in Chapter 7 *Conclusion*, together with an outlook for possible future work.

2. Background

In this chapter, related work to the topic of this report is listed and analyzed. First, some necessary foundations are looked into, which are later going to be used and applied. Then, the field of multiscale visualization is analyzed. Known techniques for multiscale visualizations are introduced and described and referencing literature is investigated. Finally, specific implementations are analyzed and their benefits and shortcomings laid out, which can later be used to improve on them.

2.1. Foundation

A **time series** is a collection of data points with a temporal dimension [15]. Single time series often are visualized in line charts, area charts, and scatterplots. Multiple time series that span an overlapping interval are the topic of various visualization techniques as they are common in many domains such as medicine, finance, and manufacturing, where they are used for analytical purposes. **Stacked area graphs** are a common approach to visualize such multiple time series [16]. Starting from a straight baseline axis (representing the time dimension), the time series are stacked on top of each other forming layers. Each layer is colored representing the series, and the thickness of the layer represents the value of the time series at each time step (also called *tick*). The resulting graph presents the evolution of the time series over time and makes it easy to read both the individual time series and the aggregated sum at the ticks. **ThemeRiver** is a technique that builds upon stacked area graphs but does not use a straight baseline axis [17]. Instead, the time series are stacked around a central baseline that lays in parallel to the temporal axis which allows the graph to have smoother transitions between the ticks. The name originates from these transitions resembling the flow of a river. **Streamgraphs** are built upon *ThemeRiver* and try to improve legibility [18]. Streamgraphs change the transition between ticks and the displacement of layers to minimize the change in slope thereby providing a more realistic presentation. The original paper provides graphical examples and detailed explanations about the difference between *ThemeRiver* and *Streamgraphs* [18].

When presenting geospatial data the goal of an intelligible visualization is to bring the data in relation to its geographic context [19]. Maps that use geographical and political features only as points of reference in the real world are called thematic maps [20]. These points have the purpose of making it easier for the spectator to connect the meaning of the displayed data to locations and regions in the real world. So-called **choropleth maps** are thematic maps that overlay the geographical or political map by coloring areas dependent on the statistical data connected to it [21]. The areas are defined by the map and not by the data that is attached [22]. The selection and application of color schemes together with the interval selection in these kinds of visualizations is housing complexity and a field of research [23, 24].

When building multiresolution maps based on geographical and political features, it is necessary to depict the given hierarchy in a uniform and structured way. The ISO published **standard 3166** defining codes for countries and partially subdivisions [25]. This standard bears the problem that it only provides full code coverage on the country level. Expanding the standard, the *Hierarchical Administrative Subdivision Codes* (**HASC**) were published

[26]. *HASC* codes provide full coverage down to the second subdivision level (e.g., in Sweden this would be municipalities) and are kept up to date by the original publisher. The codes are uniform with a length of two characters and can be chained together forming a globally unique identifier. This is important as it enables natural grouping and combining of areas based solely on their identifier. For example, all areas with an identifier starting with the code of Sweden (*SE*) together form the country of Sweden, and all areas with an identifier starting with the identifier for Kronoberg (*SE.KR*) together form the county of Kronoberg. For the municipality of Växjö this would form the identifier *SE.KR.VA* (Sweden, Kronoberg, Växjö).

2.2. Multiscale Techniques

In the field of multiscale visualization, different approaches and techniques arer esulting in substantially different visualizations as outlined in Section 1.1.2 *Multiscale Visualizations*. This section looks at different techniques relevant to this project. As the focus of this project is on techniques that work well for interactive visualizations, the selection is limited to only interactive techniques. Cockburn et al. published an excellent overview of widespread interactive multiscale techniques describing differentiations and usage examples [8].

The starting point for the following techniques is the concept of **zooming** [27]. This interaction technique is widespread and an easy utility to explore data. The idea is to allow the user to magnify (zoom in), demagnify (zoom out), and pan (moving the view) the dataset in place. In a geospatial context, an example would be Google Maps which lets the user start from an overview perspective of his area and then allows to zoom in and move the map. This concept is very versatile as it can be applied to most visualizations, for instance, most PDF-readers enable users to zoom into pages. From a multiscale perspective, this technique is not sufficient as the different scales or views are temporally separated and not displayed simultaneously, forcing the user to make the connection mentally. This puts load on the short-term memory of the user and limits the possible complexity the user is able to assimilate. The following techniques try to overcome this problem.

The first technique that improves on zooming is called ***Overview+Detail*** [8]. The idea is to provide the user with an overview of the data and, at the same time, a more detailed view into a subset by displaying two spatially-separated visualizations. One of the most commonly known examples is the usage of thumbnails. Microsoft PowerPoint, for example, shows a small list of thumbnails of the slides on the left side of the window, giving the user an overview of both the content of the slides and his position in the presentation. At the same time, the central part of the window shows a single slide in full-size giving the user a detailed view. Providing a clear connection between the overview and detail views, together with intuitive interaction possibilities, pose the main difficulty in this technique.

The next technique is called ***Focus+Context*** [9]. It is a multiresolution approach that combines focused and contextual information in a single visualization. This technique is in contrast to the previously presented spatial or temporal separation and has the potential to reduce the short-term memory load to a minimum. At the same time, such visualizations encapsulate high complexity making it harder to comprehend initially. This concept even bears high complexity in non-interactive visualizations, since uneven distortions, such as non-linear scales, are perceived as unnatural [28]. In the following paragraph, selected techniques for *Focus+Context* visualizations are introduced based on the review by Leung and Apperley [29].

For geospatial visualizations, so-called **fisheye distortions** [30, 31, 32] are sometimes used to show certain parts of a map in more detail while keeping the rest of the map un-

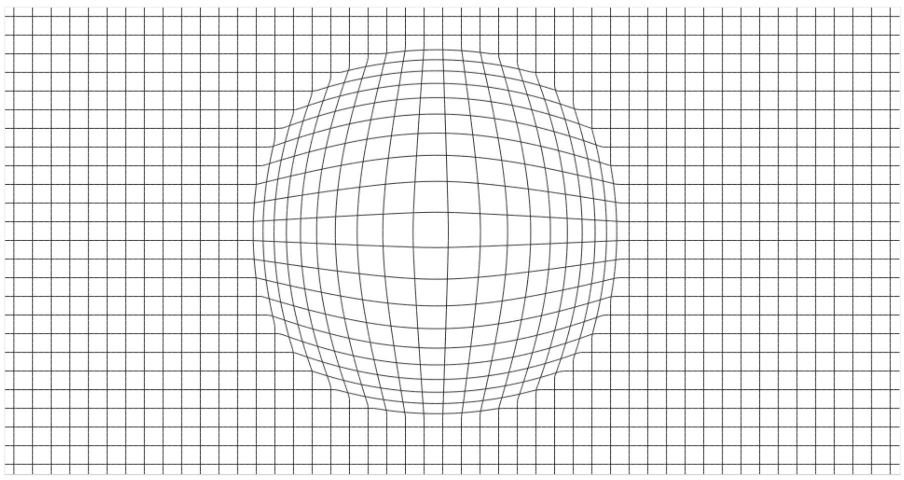


Figure 2.1.: Illustration of the fisheye distortion effect as a way to achieve a *Focus+Context* visualization.

touched. While this approach is applicable for all types of maps without further knowledge of the structure of the displayed areas, the encompassing problem is that the transitional part between focus and context is heavily distorted due to the applied polar transformation. In other words, the transformation of straight lines in the underlying map projection into curved lines make the resulting distortions hard to comprehend. An illustration of a 2D fisheye distortion with an underlying grid for better comprehension is provided in Figure 2.1. However , there is a different approach that overcomes such problems which is known as the **multilevel approach**. Here, the map is decomposed into hierarchical levels that enable the user to split up a logical, political, or geographical area into subareas. For instance, to split up a country into its composing subdivisions, which for Sweden would mean to split it up into the counties (*landsting*). This approach requires more knowledge about the displayed area but makes for an easier to understand visualization as reference points in the map are not distorted.

For time series visualizations, the situation is different as only one dimension needs to be manipulated to provide a focused view. Non-continuous magnification function can be applied without the side effect of manipulating a second, non-intended dimension. One technique is the ***Perspective Wall*** which provides a focused area—also called a lens—with a linear scale which is surrounded by two distorted scales that gradually demagnify away from the focused area [33]. Additionally, there is an adaption called ***Bifocal Display*** that removes the distortion in the context areas [34]. Instead of gradually demagnifying context areas, these are displayed as their own linear scales. These scales get chained together forming a **poly-linear scale**. This technique enables a simple to understand adjustment in granularity as the linear scales can adjust their granularity independently from each other as long as the edges follow the same principle, making the connection between scales seamless. An illustration of a poly-linear time series axis with two nested lenses can be found in Figure 2.2.

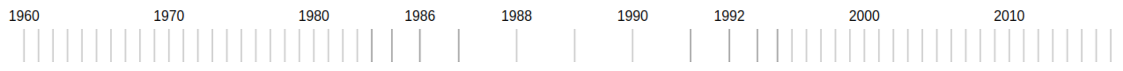


Figure 2.2.: Illustration of a poly-linear time series axis. The composing linear scales in this example are: 1960 - 1984, 1984 - 1986, 1986 - 1991, 1991 - 1994, 1994 - 2017.

2.3. Implementations

In the following section, implementations of novel multiscale visualizations and data exploration tools relevant to this project are analyzed. The goal is to understand the concepts they employ and identify the strong points and deficits of their implementation to improve on them.

MultiStream is a visualization approach to explore hierarchical time series in a multiscale way. It was introduced in a recent publication [13] and comes with an implementation and multiple examples. The approach is streamgraph-based, making it comparable to classical stacked time series visualizations. Multiple multiscale concepts were combined to form an intuitive but capable visualization. It features two visualizations in an *Overview+Detail* technique, while also utilizing the *Focus+Context* technique in the detailed visualization. To achieve this effect, the detailed visualization uses a polylinear scale forming a non-distorting lens. The centerpiece of the visualization is a multiresolution approach that splits up the hierarchical dimension inside the lens. The implementation can be found online together with multiple examples[1]. The combination of *Overview+Detail* and *Focus+Context* is made intuitive by "projection guidelines" connecting the overview and the detailed visualization both for the domain of the detailed visualization and the domain of the lens. Handles on the overview visualization allow for easy manipulation of the domains but result in a perceivable lag. These controls feel intuitive but do not allow for a change of the range of the lens. A change in the range is only possible with numerical parameters outside of the visualization that influences the ratio of the domain (the timeframe) to the range (the displayed width) and the transitioning areas. Furthermore, the approach allows for only a single lens. The resolution changes only apply to the hierarchical dimension and not to the time dimension which could cause scalability issues on larger timespans. Overall the concept and implementation are intuitive and powerful, but the focus on hierarchical dimensions cause some deficits for more generic applications.

StanceXplore is a recently presented exploration tool for *Digital Humanities* [35] that enables multi-dimensional analysis [36]. While the goal was to develop a tool for an "interactive exploration of stance in social media" [36] the resulting frontend is a more generic framework for analyzing multi-dimensional datasets. This tool is particularly interesting for this project as it has the same target group. The core concept of the presentation is the usage of coordinated views. Every view represents a dimension and offers filtering options that are reflected in the other views. This concept allows the user to filter in multiple dimensions while keeping an overview over all dimensions. The concept also includes views for geographical and temporal distribution and the time series features an *Overview+Detail* technique. The geographical view includes a multiresolution map that uses temporal separation. The map view enables both the visualization of the county and the municipalities level. However, this multiresolution approach is only available for all counties at once. The coordinated views make interacting with the data intuitive, but for the web unusual key combinations could cause context disruptions. The multiscale time series has no option to filter the dataset, and the selection of the domain for the detailed visualization is unprecise hindering the usability and utility of the view. An implementation with the example of stances in Twitter data that runs in the browser exists online[2], and an official demonstration video was published[3].

[1]http://advanse.lirmm.fr/multistream/ [Accessed: 23-Apr-2018]
[2]http://sheldon.lnu.se/stancexplore/ [Accessed: 22-May-2018]
[3]https://vimeo.com/230334496 [Accessed: 23-Apr-2018]

3. Methodology

This chapter describes the research methods used to find answers to the research question. First, the used methods are listed and introduced. Then, the reliability and validity of both the methods and the project as a whole are discussed, and finally, ethical considerations are laid out.

3.1. Scientific Approach

A combination of different research methods was applied to draw scientific conclusions and find valid answers to the research question. As an overall strategy, the method of *Verification and Validation* was used. This structured the project in three phases: (i) the requirement collection and conception of the system, (ii) the implementation phase, and (iii) the evaluation phase, consisting of a verification and a validation part.

Other methods were used inside this structure such as a case study, an experiment, and a survey and open expert interview. A more detailed explanation of these methods and the way they were performed can be found in the next section. An illustration of the applied methods structured under the overall strategy can be found in Figure 3.1.

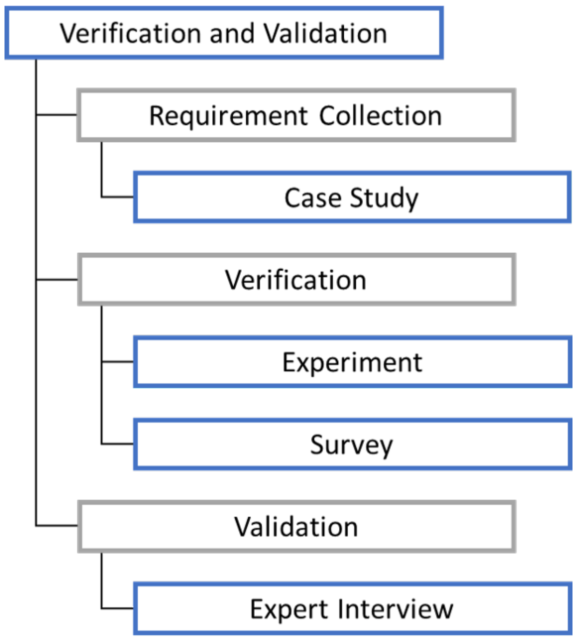


Figure 3.1.: Overview of the structure of research methods applied in the thesis. Research methods are highlighted in blue, while method parts are in grey.

3.2. Method Description

The first step in the chosen strategy is to perform a comprehensive **requirement collection** for the planned system. The goal is to make sure that all aspects of the research question are met, and nothing important is forgotten along the way. The list of requirements, together with justifications and clarifications for every requirement, can be found in Section 4.1. In the evaluation phase, the Section 6.3 ties everything together by going over every requirement again and making sure they were fulfilled.

To be able to evaluate the work and to conduct the following research methods, the system had to be implemented in specific scenarios. For this reason, two **case studies** were conducted.

The first case study was designed with the purpose of showing the extent of the work within a realistic scenario, based on an existing system, and was therefore used for all evaluation processes and presentations. This scenario was built on top of a collection of Scandinavian Tweets. A tweet is a public message on the short messaging platform Twitter. This dataset can provide insights into the usage of language in Tweets from a timespan of over a year which could be of interest to humanity researchers. A detailed description of the scenario can be found in Section 4.3.1, and the work necessary for the implementation within the system is described in Section 5.3.1.

The second case study was designed with the purpose of highlighting the ease with which new datasets and scenarios can be plugged into the system. Therefore, a dataset from Södra–one of the largest forestry companies in the south of Sweden–was chosen which has the potential to provide insight into forestry yields of different woodlands they cultivate. The full description of the case study can be found in Section 4.3.2, and the implementation in Section 5.3.2.

In the evaluation phase, at first, the results were verified to check if the requirements were met. To come to a scientific conclusion, two different quantitative research methods were used in the verification part: an experiment and a survey.

An **experiment** was conducted to measure the performance of the system in different aspects and for different data sizes and tasks. The target was to check if the system can handle high loads and how it performs with generic tasks. The setup of the experiment is described in Section 6.1.1, and the findings are reported in Section 6.2.1. The measured metrics include, among others, the Time-to-Interactive, transmitted dataset size, and the memory usage in the browser. Additionally, the performance of *StanceXplore* was measured for specific tasks, and a rough comparison was drawn between the performance of both systems. Together with a feature comparison in regards to the scope of the system, this allowed for an overall comparison between the systems.

A **survey** was conducted with all attendees of the later on described expert interviews and consisted of a single questionnaire, the *User Experience Questionnaire* (UEQ). It is a widely used questionnaire to measure user experience (UX) that was initially created in 2005 by researchers at the SAP AG [37]. Today the *UEQ* is available in multiple languages and comes with prepared analysis tools, and a large comparison dataset. The consistency of the scales and their validity was confirmed in multiple studies and the comparison dataset contains reference values from over 160 studies with over 4800 polled individuals [38].

In 26 contrastive pairs, the survey tries to measure the general impression of a user regarding his interactions with an interactive product [37]. The 26 items are seven-stage Likert scales between semantic differentials which can be grouped up to the six scales: (i) *attractiveness*, (ii) *perspicuity*, (iii) *efficiency*, (iv) *dependability*, (v) *stimulation*, and (vi) *novelty*. Perspicuity, efficiency, and dependability describe pragmatic qualities, while stimulation and novelty describe hedonic qualities. Attractiveness exists independently as a pure valence dimension. The structure of the questionnaire makes it easy to spot inconsistent and suspicious answers while showing strengths and weaknesses of the system thanks to the reference values.

The second part of the evaluation phase is the validation, where the overall concept and the chosen solutions were checked against the expectations of possible users and people with domain knowledge. To get this qualitative feedback, an **open expert interview** was conducted with five experts in the field of data visualization and data exploration. In this process, the interviewee was given a short introduction to the system and its usage, and then he was asked to perform a few tasks with the system in a recorded *think-aloud* session. At this point, the user was asked to fill out the previous mentioned *UEQ*. After that, an open interview was conducted to get feedback on the used concepts, the usability and understandability, and generic improvement possibilities. The exact procedure and its results can be found in Chapter 6.

3.3. Reliability and Validity

Some aspects of the project make it necessary to discuss validity and reliability. Regarding *reliability*, the measured metrics in the experiment relied heavily on many factors of the test environment, from operating system to clock speed, both on the client and on the server. These factors were locked down and documented, which made the experiment reproducible. Another less-important factor was the fragmentation of the dataset concerning specific filters and test set sizes. This influence was smoothed out by the newly-shuffled test set in every iteration. All details regarding the setup of the experiment are explained in Section 6.1.1.

In the aspect of *validity*, there were the results of the experiment, survey, and the expert interview to consider. The experiment results should be considered valid since they only compared own measurements from the same system and do not draw any conclusion based on outside measurements, and only support observations made from the interview and survey. The survey and the expert interview might be prone to problems with internal and external validity. Internal validity is about drawing the right conclusions from the collected data and external validity is about the extent to which a result can be generalized to more generic conclusions. The survey and the expert interview were conducted with only five participants of which none were in any way connected to the project, but all of them came from (or are very familiar with) the field of visualization. To overcome these problems Chapter 7 makes extensive efforts to clarify that the results from these two methods are not universal, only providing initial and implementation-specific insights and feedback.

3.4. Ethical Considerations

It is important to keep ethical considerations in mind when dealing with human subjects in any project. In this project, the only ethical considerations come from privacy and data protection originating from the datasets used in the case studies and the recorded data from the expert interviews.

While the dataset from Södra does not contain any personal information, the dataset from Twitter does contain public data from users, and the pseudonymized usernames and the contents of the tweets themselves may possibly contain personal data. In their privacy regulations and developer guidelines[1], Twitter specifies that the data associated with tweets are free to use as long as the guidelines are followed. The end user has the possibility to opt-out of this data sharing, and has to opt-in for location sharing as a unique form of personal data sharing.

[1]https://developer.twitter.com/en/developer-terms

For the interviews, the participants were informed that no private personal information was recorded and all of the findings would only be published independently from their person, and on an aggregated basis. With the agreement of the interviewees, the whole process of software trial and the interview itself was recorded to ensure that nothing important was lost in the process. These recordings were only stored for this process and without any personal pieces of information attached (although a connection could be made through the interviewer because of the small sample size). After the evaluation process ended, all recordings and raw notes on specific interviews were destroyed.

4. Conception

In this chapter, the concept of the system is described. First, the requirements collected for the system are detailed and discussed. Building upon these requirements, the design of the system is described and, finally, two case studies are conceived and explained.

4.1. Requirements

The first step in the *Verification and Validation* method is to compile a list of both functional and non-functional requirements as explained in Section 3.1. The following requirements are built around the idea of a hierarchical organization, beginning with a few high-level non-functional requirements that dictate how we want the final system to behave. Under these main requirements are the sub-requirements that, when fulfilled, should satisfy their parent requirement. This way, low-level sub-requirements can be formulated more technically, making them clearer to implement, while we still maintain the high-level view of the main requirements.

To make the system practically usable, we wanted to make it as *accessible* as possible. This required: (i) keeping the initial setup for new users as easy as possible, (ii) decoupling the presentation from the dataset, and (iii) making the presentation accessible with tools that most users are expected to be familiar with.

Requirement 1 (Accessibility)
The system has to enable multiple users/researchers to look into the same dataset at the same time, without having to set up the system multiple times.

Requirement 1.1
The presentation has to be accessible across different devices without unnecessary overhead.

Requirement 1.2
The presentation has to be decoupled from the data source.

One of the key features of the system was defined to be the *adaptability* to different datasets, providing an added benefit that exceeds its usage in a single installation. To achieve this, we had to require adaptability both in regards to data sources, data structures, and viewing angles, while keeping the effort to set these up to a minimum.

Requirement 2 (Adaptability)
The system has to be easily adaptable to different datasets regarding structure, data source, and viewing angle.

Requirement 2.1
The presentation has to be easily configurable to provide different views into a dataset.

Requirement 2.2
The system has to be easily adaptable to handle new data sources and data structures.

Requirement 2.3
The user has to be able to quickly swap between different setups.

When providing different viewing angles into a dataset, it is important to bind them together to make a single, conform presentation. Therefore all viewing angles were required to be *filterable* while keeping them connected and making the connection apparent to the user.

Requirement 3 (Filterability)
The user must be able to filter the dataset from a selected viewing angle in the presentation, according to the view's unique presentation features.

Requirement 3.1
A filter applied in one viewing angle must be broadcasted directly to all other viewing angles.

Requirement 3.2
The user has to be able to filter the dataset down to a single event.

One key research focus of this project was to bootstrap the underlying framework for the creation of meaningful *multiscale* visualizations in multiple views of a multi-dimensional dataset, and embed them seamlessly in a presentation.

Requirement 4 (Multiscale)
The presentation has to provide the possibility of visually exploring datasets in multiple scales in every viewing angle that would benefit from it.

Requirement 4.1
The presentation has to provide a map view that employes multiscale techniques.

Requirement 4.2
The presentation has to provide a stacked time series view that employes multiscale techniques.

ISO 9421 *Ergonomics of Human System Interaction* [39] in its 2018 revision split up part 11 which previously defined specifications and measurements of usability. As this project and the requirement collection began before the publishing of the 2018 revision, the requirements were designed along the previous versions part 11, *Guidance on usability* [14]. Research suggests that creating requirements along the guidelines and definitions of this standard is effective, but very hard to do if the requirements are applied in details [40]. Therefore, the derived requirements only take the standards as guidelines, and not as strict rules. Although most previously described requirements can be factored into the *usability* of the system, some additional requirements were needed.

Requirement 5 (Usability)
The system has to follow general guidelines for usability.

ISO 9421-11 defines three guiding criteria for the usability of software: the effectiveness of task completion, the efficiency of usage, and the satisfaction of the users. The standard defines efficiency as "the resources expended in relation to the accuracy and completeness with which users achieve goals." [14] This means that the user has to be able to solve a task with a system with as little effort as possible. To achieve this, the user interface should be kept as straightforward as possible to keep the focus on the data and its expressed meaning.

Requirement 5.1
The presentation shall not include any unnecessary information or possibilities of interaction.

Additionally, it is important to keep the user focused on his tasks without breaking his “train of thoughts”. To achieve a context-disruption-free presentation, an important requirement is to keep the amount of data showed in the presentation itself to a minimum.

Requirement 5.2
The presentation shall avoid context disruption on interactions as much as possible.

Requirement 5.3
The amount of transfered data from the data source to the presentation shall be kept as light as possible.

Effectiveness is defined as "the accuracy and completeness with which users achieve specified goals" by the ISO standard, which means that the system should provide the possibility to solve tasks as completely and correctly as possible [14]. Effectiveness is linked to the satisfiability of the previous requirement. Lastly, satisfaction is defined as the "freedom from discomfort and positive attitude to the use of the product." It is the most comprehensive criteria and includes points such as learnability, perspicuity, and general attractiveness. The target was to project on the user a coherent, pleasant overall impression that illustrates the utility and added benefit of the system.

Requirement 5.4
The presentation shall employ different, clear to understand colorschemes.

Requirement 5.5
The added value of the system as a whole for the data exploration shall be clear to users.

Requirement 5.6
The added value of the employed multiscale techniques shall be clear to users.

To guarantee an apparent learning curve and perspicuity of the users’ own actions, some additional requirements were necessary.

Requirement 5.7
The basics of interacting with the data through the presentation shall be easy to learn.

Requirement 5.8
Every possible action by the user shall be reproduceable for him.

Requirement 5.9
The user shall be enabled to solve possible tasks with the system after a short introduction into the presentation.

A full list of all requirements can be found in Appendix C.

4.2. System Design

In this section, the design of the system itself is described (not its implementation, which is described in Chapter 5). It incorporates the requirements into an overall structure geared towards their realization. At first, the architecture is described with all of its components and their connections and couplings. Next, the user interface and in particular the visualizations are looked into in more detail as the emphasis of this project is on the multiscale data visualization.

4.2.1. Architecture

To make the system conform to the accessibility requirements (Req. 1) a classical client-server architecture was chosen. The server-side handles the datasets, configurations, and data preparations, while the client takes care of the presentation (Req. 1.2). To make the presentation accessible on most devices without any further installations the browser was chosen as a client (Req. 1.1).

The adaptability requirements also heavily influenced the overall architecture. To make new datasets and viewing angles easy to set up without setting up the whole service the server-side had to be split up into the main backend service and connectors querying the data source representing use cases.

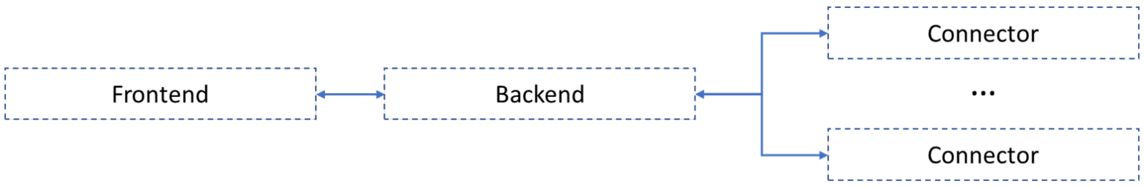


Figure 4.1.: High-level overview of the components of the system.

This leaves the system split up into (i) the frontend, (ii) the main backend service, and (iii) connectors as visualized in Figure 4.1.

The **backend service** (Figure 4.2, middle) is designed to function as a server for the frontend and as an *application programming interface* (API) gateway between presentation and data source. The central component has to be the configuration controller that handles the configuration of every view into the corresponding dataset. When a request reaches the backend service, it shall resolve the origin of the request and look up the corresponding configuration file. If there is a valid configuration for that host, the request itself is looked at. If it a request to serve a presentation the static frontend is served. If it is a configuration request from a presentation, the configuration is pulled from the internal configuration store and served. If it is a request for data, the service queries the connector and then aggregates and compresses the data to transmit as little to the presentation as possible (Req. 5.3).

The **frontend** (Figure 4.2, left) is providing the presentation, and the design is described in detail in the following subsection. The frontend was designed to be completely static, meaning that the same frontend is used for all presentations regardless of dataset and viewing angles. Only after the frontend is served, it shall request its configuration and adapt the presentation accordingly. Not having to adapt the frontend also helps with the adaptability requirements.

By having a single backend service and a static frontend utilized in all use cases, the user is able to switch between already set up presentations by simply navigating to the *Uniform Resource Locator* (URL) that hosts the desired presentation (Req. 2.3).

The core of every use case has to be the **configuration**. The connectors register themselves at the backend service by submitting their configuration. The configuration was designed to always provide the following information: (i) the *URL* of the connector, (ii) the name, (iii) the hostname, (iv) the dimensions of the dataset and their type, (v) starting filters, and (vi) the presentation blocks setup. The *URL* is needed for the backend to reach the connector upon queries and the hostname is required to identify the corresponding connector upon requests. The name is thought of as an identifier in the presentation and is passed to the frontend together with initially applied filters and the setup of the presentation blocks. Every block configuration has to include a title, the type of the block and a width. Additionally, further information may be required depending on the type of the block. At a minimum, this includes the displayed field. The presentation blocks themselves are described in the next subsection. The backend shall store all registered configurations locally for quick access and maps the incoming requests to connectors using the hostname in the configurations. By changing the configuration or submitting a new configuration using an already existing connector, it should be easy to set up new viewing angles into a dataset (Req. 2.1).

The **connectors** (Figure 4.2, right) are the only part of the system that has to be adapted to new data sources and structures. Their only required function is to query their data source when the backend service requests a set of data. To connect backend and connector, an endpoint is required that takes in a set of filters and fields. In the first iteration, these filters are defined as *startsWith*, *between*, *excludes*, and *includes*. These filters are known in most database querying languages and should be easy to implement for the specific data source. The main idea behind the connectors is that most of the calculation effort is taken care of by the database which is optimized to do exactly such tasks. This design keeps the required code to set up a new data source and structures to a minimum as required by Req. 2.2.

In conclusion, the architecture consists of a single static frontend and a backend service that are populated by the data provided by connectors that are purpose build to handle their data source (Figure 4.2). By quickly adapting the connector a researcher is able to set up a new data source and by changing a simple configuration providing different views into this dataset. Once this is done the presentation is accessible to everyone with access to the hosted service by using their browser.

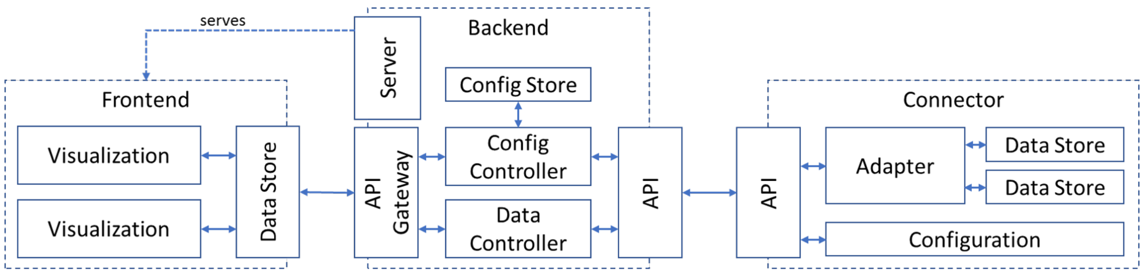


Figure 4.2.: Illustration of the components and subcomponents of the system.

4.2.2. User Interface

The frontend was designed to be static so that it could be used for all use cases and only adapts at runtime to the configuration of the current presentation. To achieve this, the frontend shall provide multiple components that can be used to provide a viewing angle into the data. The components shall be easily adaptable in order, size and of course displayed dimension of the dataset by the configuration of the presentation.

For the prototype, the following components were conceived: (i) a plain list, (ii) a category list, (iii) a time series, and (iv) a map.

The **plain list** was thought of as the simplest component that only takes a dimension of the dataset and asynchronously loads a configured amount of entries to display. Filters shall be applied, but the component itself shall not provide any mean to filter.

The **category list** has the purpose of filtering the dataset in a categorical dimension while visualizing the magnitude of events between the categories. In contrast to the plain list the data is not separately loaded, but a dimension of the global dataset is used. Every list entry shall represent a category and consist of a tick box for filtering, the title of the entry, a bar representing the percentage of events to the biggest category, and the total event amount. To make the list easier to scan the categories shall be color-coded following Req. 5.4. Additionally, the list shall be ordered descending by the number of events. A mockup can be seen in Figure 4.3.

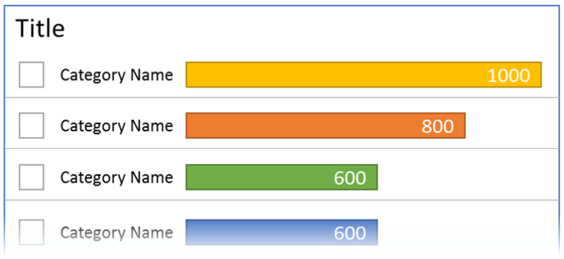


Figure 4.3.: Mockup of the category list component.

The next component is the **time series** component, designed to display event an event count split into a categorical dimension over a time dimension. The component shall enable the user to filter the dataset in the time dimension and provide a seamless multiscale experience over the same dimension. The multiscale concepts chosen for this are described in more detail in Section 4.2.4. As this is visually a very complex component, it was important to keep Req. 5.1 in mind to avoid context disruptions through confusion as required by Req. 5.2. By displaying categories which are possibly already displayed in a category list, it was important to remember to keep the used color-coding for the categories consistent. Apart from the multiscale visualization, this component was not designed to feature any other functionalities.

The **map** component is the second component featuring a multiscale visualization. This component shall take a single hierarchic categorical dimension to aggregate and display the event count in the provided map material. Based on this approach of utilizing custom map material, not only geographical maps but also thematic or abstract maps are possible, making this component highly flexible and powerful. A detailed description of the multiscale approach for this visualization can be found in the next subsection (Section 4.2.3). Similarly to the time series component, this component was designed to only feature a single visualization to keep the complexity low (Requirements 5.1 and 5.2).

To tie all designed components together in a presentation and provide meaningful exploration functionalities, it was necessary to come up with a combined filtering concept that can handle the multi-dimensional nature of the datasets (as required by Req. 3). The goal was to provide filtering options in one dimension while reflecting the applied filters in all other dimensions in the visualization of the current dimension. For example, only showing categories that have events in the current selection, and omitting categories that have no events, since filtering on these would have no impact on the other dimensions. To achieve this, all filtering components shall propagate filtering events to the central data store which then shall apply the filter (and fracture the dataset) to all other dimensions. This is necessary because we also want to show the excluded values in our filtering dimension. This approach ensures that a filter applied in one viewing angle is directly propagated to all other viewing angles as required in Req. 3.1. Moreover, the overlapping filtering across dimensions, enables the user to filter down to a single event if it is unique in its combination of used dimensions (Req. 3.2).

4.2.3. Map Visualization

The map component shall feature a single multiscale map as required by Req. 4.1. To make the map flexible, the configuration has to provide links to map material, which includes the shapefiles plus level identifiers. Shapefiles contain the necessary contours of the areas and the level identifier make these areas destinctly identifiable.

The basic concept of this visualization is a choropleth map, i.e., to draw the shapes of areas and to fill it with color based on the count in this area on a quantized scale. To map entries to areas, it was required to map the areas into hierarchical levels. For example, Växjö is a municipality in the county of Kronoberg, which is part of the country of Sweden. The multiscale approach is not to display just one level (i.g., municipalities, counties, or countries) at a time, but to give the user the possibility to split up specific areas into its subareas. To come back to the example, this would mean that the user can look at Växjö and all other municipalities in Kronoberg while, at the same time, compare them to the surrounding counties. A mockup of this example can be seen in Figure 4.4. This hierarchical approach allows the visualization to combine subareas to larger areas (i.e., municipalities to counties) by summing up all subareas contained in the area. It is important to note that this visualization is not limited to geographical maps but can also be used in abstract maps as the shapefiles are provided by the configuration.

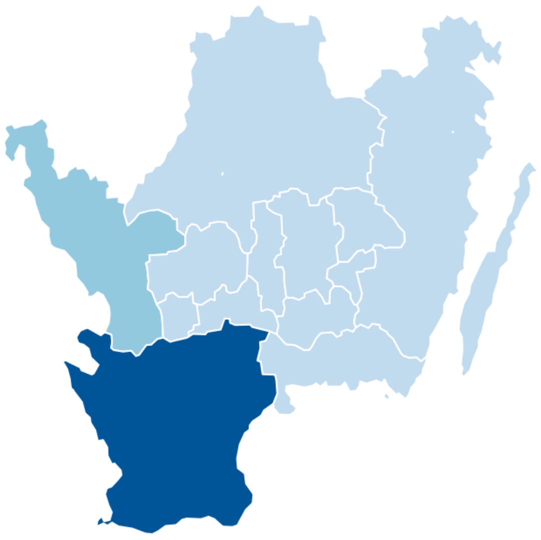


Figure 4.4.: Mockup of the map visualization showing the county of Kronoberg split into municipalities, while the surrounding counties are intact.

Additionally, the map needs further features such as a color legend, zoom and pan controls to navigate through the map easily, and controls to filter for specific areas.

In conclusion, the map visualization was designed to be a choropleth map that allows the user to split up areas into subareas and thus change the granularity of specific regions while keeping the rest unmodified at an overview level.

4.2.4. Time Series Visualization

The time series component shall feature a stacked time series visualization as required by Req. 4.2. To accomplish that, the visualization shall consist of three parts: (i) an *overview time series*, (ii) a *detailed time series*, and (iii) a control layer connecting the two time series.

The overview time series shall be a small stacked area chart over the full timespan of the targeted time dimension. The purpose was, as the name suggests, to provide an overview over the whole timespan and enable the selection of the domain of the timespan of the detail view. As the y-Axis stacks counts of categorical data, it is important to keep color-coding consistent not only with the detailed time series but also to possible other representations of this dimension in the current overall presentation. Because the purpose of this time series is only to provide an overview, the granularity in the time dimension is only of little importance and can be as fine as the data is compacted, to keep calculation overhead low.

The detailed time series shall visualize the selected domain in more detail. This shall be achieved by stretching the domain (the timespan) over the full width of the visualization giving it a wider range (the displayed width). Additionally, a *Focus+Context* approach shall be embedded, allowing the user to go into even more detail without losing the context of the rest of the detailed domain. Furthermore, the user shall be able to add lenses with both a variable domain and range. This enables the user to dynamically focus on a specific timespan and stretching it to a width that allows the granularity he desires. As this approach makes the horizontal axis polylinear, without any restrictions on how far a timespan will be stretched or squeezed, the granularity of every stretch of the axis has to dynamically adapt based on its range.

The control layer shall connect the two time series and give the user the handles to select the detailed domain, add lenses, and manipulate their domain and range. To achieve this, the control layer shall add handles to both time series which are visually connected to their counterpart on the other time series. The handles on the detailed time series shall enable the user to adjust the range of a lens and the handles on the overview time series to adjust the domain.

In conclusion, the time series visualization is a combination of the *Overview+Detail* and the *Focus+Context* technique with a dynamic selection and focus, as illustrated in Figure 4.5. This concept was influenced by the *MultiStream* visualization introduced in Chapter 2 *Background* while introducing more intuitive and dynamic controls and a changing granularity not in the category dimension but the time dimension.

4.3. Case Studies

For the evaluation, the system was implemented for specific scenarios. Hence, these two scenarios are explained and discussed.

4.3.1. Twitter Language Usage

The scope of the first case study was to embed the system in a realistic scenario. For that purpose, the following scenario was chosen: As a researcher, I want to investigate the

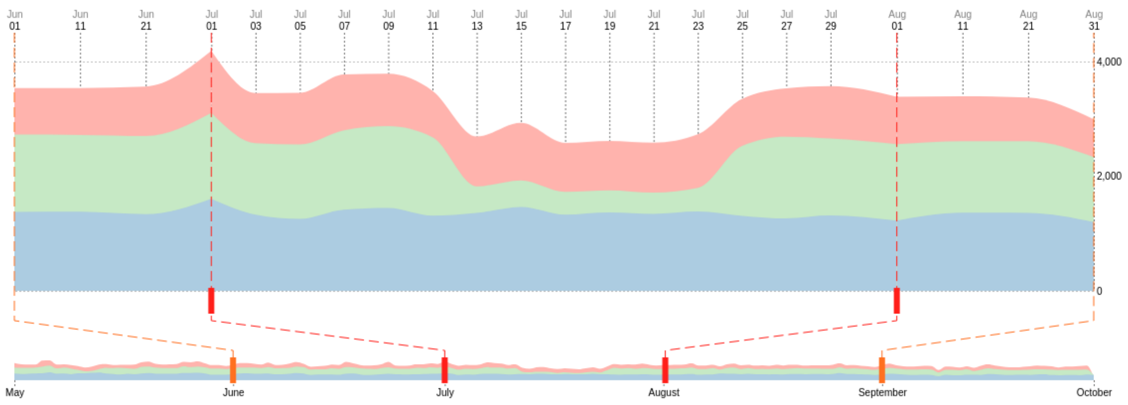


Figure 4.5.: Mockup of the time series visualization showing a detailed view from June to September and a focus lens for the month of July.

usage of different languages in Tweets in the different regions of Scandinavia and how they develop over time.

This case study is based on the Nordic Tweet Stream (NTS) [41]. The dataset includes 11 million Tweets from Denmark, Finland, Iceland, Norway, and Sweden over a timeframe of 15 months starting in November 2016. For this case study only the origin, the timestamp, language, and content of the Tweet are of relevance. Additionally, the dataset contained mostly meta data about the poster and the location.

Based on this dataset there already was a tool with a similar use case developed, which was introduced in Chapter 2 *Background*. This tool was adopted as a benchmark with the goal of comparing to the system conceived here regarding performance and general feature richness. The comparison is captured in an experiment that is detailed in Section 6.1.1 *Experimental Setup*.

To realize this scenario, the system shall be configured to display the following components with their respective dimensions. Firstly, a category list representing the language dimension enabling the user to compare used languages in a fixed timeframe and filter down on specific languages. Next, a plain list of Tweet contents to get a feeling of the Tweets themselves. Then, a map component with the Scandinavian countries and their different administrative levels to compare and filter down on regions. Lastly, a time series component visualizing the development of the languages over time.

4.3.2. Södra Forestry Yields

The second case study has the goal of highlighting the ease with which new datasets and scenarios can be plugged into the system, as required by Req. 2.

To achieve this, the case study was based on a different dataset. The utilized dataset comes from Södra[1] and reports the yield of harvested trees including timestamp, location, and species. The chosen scenario is the following: As an operator, I want to know how many trees per species were cut down in different areas over time.

For this scenario, the presentation shall be configured to display the species in a category view, and the positions on a map of the south of Sweden. Furthermore, the time series component had to display the species over time. The data source for this scenario should be different from the previous one to give another example implementation and demonstrate the ease of adaption.

[1]https://www.sodra.com/en/ [Accessed: 06-Mai-2018]

5. Implementation

After the conception phase, the concept had to be implemented to prove its viability and enable the testing of the case studies and further evaluations. In the following sections, the implementation of the backend service, the frontend, the visualizations, and finally the connectors for the two case studies are briefly described, and problems during implementation are laid out. The source code of all components can be found online in their corresponding repositories [1].

5.1. Backend

The backend service was implemented in JavaScript utilizing the *node.js*[2] runtime, which is powered by Google's *V8 engine*[3] that also powers the *Google Chrome* browser. *Node.js* is available for all major server operating systems and enables researchers to painlessly set up their own instance locally or on a server. By using JavaScript in all components of the system, it was possible to share code between the components and lower the hurdle for further development. The backend service utilizes the lightweight *express framework*[4] to provide a server and handle requests. *Node.js* is a JavaScript runtime that elevates the language from a purely browser based one to a language that is able to produce standalone applications and webservers. The *express framework* is a wrapper around the *node.js* webserver protocols that aides in routing and request handling. All used technologies and tools are open-source and freely available.

The backend service requires three controllers: (i) a server controller, (ii) a configuration controller, and (iii) a data controller.

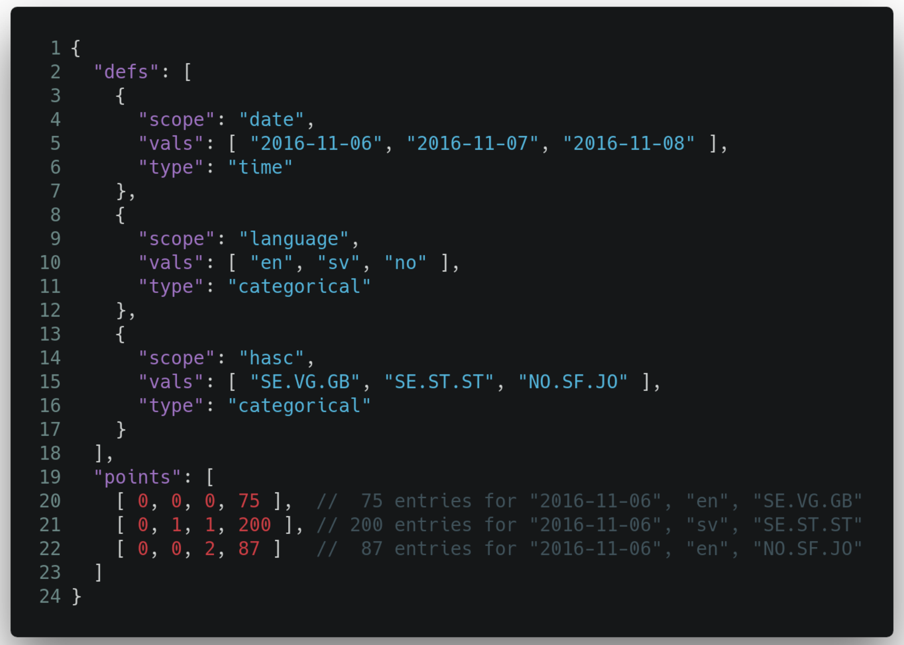
The **server controller** serves the files of the static frontend in production mode or functions as a proxy in development mode. With only 12 lines of code, this is by far the smallest controller.

The **configuration controller** takes care of registering and unregistering use cases and their connectors. To do this, it provides two endpoints: (i) a registration endpoint and (ii) an unregistration endpoint. The former takes the submitted configuration and looks up if the use case is already registered by checking the hostname, and either updates the entry or adds a new one. The latter does exactly what the name says and discards the matching entry in the local store. Furthermore, this controller provides functionalities to the other controllers by providing wrappers around finding and listing stored configurations. The configurations themselves are *JavaScript Object Notation* (JSON [42]) objects that implement the schemata introduced in Section 4.2.1 *Architecture*. An example of a configuration can be found in Appendix A.

The **data controller** takes care of data requests from the frontend. To achieve this, it looks up the configuration file for the requested hostname and then queries the corresponding connector with the required fields, filters, and limit. The returned events are then compacted and compressed before they are sent back to the requesting client. The

[1]https://gitlab.com/isovis/stancexplore
[2]https://nodejs.org/ [Accessed: 06-Mai-2018]
[3]https://developers.google.com/v8/ [Accessed: 06-Mai-2018]
[4]https://expressjs.com/ [Accessed: 06-Mai-2018]

compacting of the data is achieved by building a cross product of all requested dimensions and counting the events for each combination. This splits up the response in (i) a *definition* collection, listing all dimensions with their type and all occurring unique values in an array, and (ii) a *value* part, listing all occurring combinations together with their absolute frequency. Each *value* entry is an array, with the last index being the count. Each entry in this array is mapped by its index to the dimension in the *definition* part, and its numeric value to the index in the value list of the dimension definition. An example is illustrated in Figure 5.1. This way the dataset is compressed for transmission without losing any important information. This self-defined concept could be further improved by adapting the definition part of dimensions based on their type. For example, a time dimension could provide its starting interval, and the referenced index could be calculated by counting the intervals after that without the need to explicitly list them.



```
{
  "defs": [
    {
      "scope": "date",
      "vals": [ "2016-11-06", "2016-11-07", "2016-11-08" ],
      "type": "time"
    },
    {
      "scope": "language",
      "vals": [ "en", "sv", "no" ],
      "type": "categorical"
    },
    {
      "scope": "hasc",
      "vals": [ "SE.VG.GB", "SE.ST.ST", "NO.SF.JO" ],
      "type": "categorical"
    }
  ],
  "points": [
    [ 0, 0, 0, 75 ],  //  75 entries for "2016-11-06", "en", "SE.VG.GB"
    [ 0, 1, 1, 200 ], // 200 entries for "2016-11-06", "sv", "SE.ST.ST"
    [ 0, 0, 2, 87 ]   //  87 entries for "2016-11-06", "en", "NO.SF.JO"
  ]
}
```

Figure 5.1.: An example of the output of the data compacting algorithm. The first line in the *points* array expresses that there are 75 entries with *2016-11-06*, *en*, *SE.VG.GB*.

In conclusion, the backend service is lean and easily extendible to provide additional features down the line. It needs no code changes to serve any use cases and in theory, can facilitate unlimited use cases. As the only calculation heavy functionality is the compacting of requested datasets, scalability on a single process should be secured even under heavy usage. Extending this usage barrier the service is also able to seamlessly run in a cluster mode managed by an external process manager like pm2[5] making it as scalable as the environment it is hosted on by spreading requests between multiple instances.

[5]http://pm2.keymetrics.io/ [Accessed: 06-Mai-2018]

5.2. Frontend

The frontend is also a JavaScript project that relies heavily on build tools to provide the best cross-browser compatibility and smallest file size while using the newest language features. *Babel*[6] and *webpack*[7] are used in the build process while the bundled output only packs six external libraries. *Vue.js*[8] is used as a reactivity and display framework with *Vuetify*[9] as component and styling framework on top. This makes clean, encapsulated components possible helping with code sanity, extendability, and maintainability. *Axios*[10] is used as an HTTP client for asynchronous requests and *lodash*[11] as a high performant utility library. Additionally, the visualizations are built upon *D3*[43] and *TopoJSON*[12] which enable building powerful visualizations from scratch. Again all used technologies and tools are open-source and freely available.

The structure of the frontend is straightforward. The core takes care of the configuration and data fetching and storing while all other functionalities are encapsulated in separate files. Apart from the different components, there is a parser taking care of all data crunching and a profiler for evaluation reasons.

Upon loading, the core requests the configuration and initial dataset from the backend service. Once the configuration is loaded, the presentation is built by appending the defined components to the component list. The components are provided with their configuration and the linked data collection. Additionally, the title of the presentation is inserted and the initial filters are set. When the dataset is loaded, the parser builds a full collection out of the compacted data and defines the colors of every category across all dimensions. After the data is loaded and transformed, the collection is split up into its dimensions and filtered according to the set filters. The filtered collections are then fed to their linked components which update their visualization.

To set up the components, the order in the definition is used to sort them and the provided width is projected on a grid of 12 units. The list is wrapped, forcing components in a new row once all 12 units of a row are filled, resulting in a distribution of the components across the screen that is easy to define and understand.

When a component registers a filter change by the user, the new filter is propagated to the core which applies the new filter to all other filtered collections, resulting in their displaying components to dynamically update their visualization.

The components are split and encapsulated in their own directories and files, making them easily modifiable and extendable without influencing any other component. While the data transformation is taken care of in the parser, minor adjustments that are not generalizable are housed in their component, keeping the parser clean. All components take their title, linked field, filtered collection, and applied filter as properties, making them fully reactive to changes applied by the core.

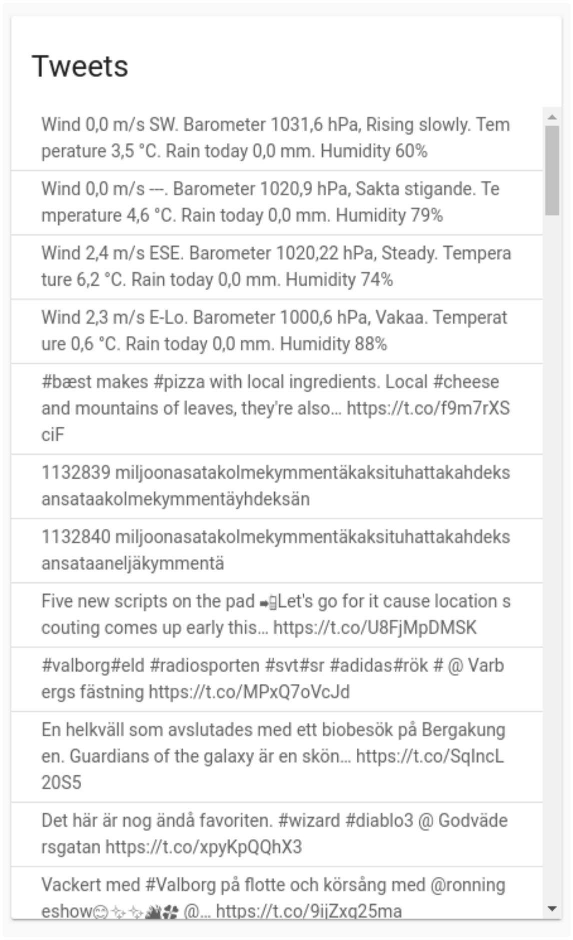
In conclusion, the structure of the frontend is highly abstracted and encapsulated, making it easily extendable with additional components and making the components flexible to handle different datasets. With the configuration file, the presentation is easily configurable by chaining components together and binding them to their data dimension. The whole frontend project is static and can be used for all presentations as it adapts to the

[6]https://babeljs.io/ [Accessed: 06-Mai-2018]
[7]https://webpack.js.org/ [Accessed: 06-Mai-2018]
[8]https://vuejs.org/ [Accessed: 06-Mai-2018]
[9]https://vuetifyjs.com/ [Accessed: 06-Mai-2018]
[10]https://github.com/axios/axios/ [Accessed: 06-Mai-2018]
[11]https://lodash.com/ [Accessed: 06-Mai-2018]
[12]https://github.com/topojson/topojson/ [Accessed: 06-Mai-2018]

configuration at runtime.

5.2.1. Plain and Category List

The plain list component was implemented as straightforward as designed. The component expects a title, the field to display, and the full set of filters. If the filters change, the component asynchronously queries the backend service and displays a loading state. Once the backend service returns the entries, they are displayed. A screenshot of the finished component can be found in Figure 5.2. This component could be easily extended to fit further purposes by expecting a template literal and a field list instead of just a single field to display custom results for each entry. For the prototype this was not done as there would be no benefit for the two case studies.


Tweets

Wind 0,0 m/s SW. Barometer 1031,6 hPa, Rising slowly. Tem perature 3,5 °C. Rain today 0,0 mm. Humidity 60%

Wind 0,0 m/s ---. Barometer 1020,9 hPa, Sakta stigande. Te mperature 4,6 °C. Rain today 0,0 mm. Humidity 79%

Wind 2,4 m/s ESE. Barometer 1020,22 hPa, Steady. Tempera ture 6,2 °C. Rain today 0,0 mm. Humidity 74%

Wind 2,3 m/s E-Lo. Barometer 1000,6 hPa, Vakaa. Temperat ure 0,6 °C. Rain today 0,0 mm. Humidity 88%

#bæst makes #pizza with local ingredients. Local #cheese and mountains of leaves, they're also... https://t.co/f9m7rXS ciF

1132839 miljoonasatakolmekymmentäkaksituhattakahdeks ansataakolmekymmentäyhdeksän

1132840 miljoonasatakolmekymmentäkaksituhattakahdeks ansataaneljäkymmentä

Five new scripts on the pad ➡📱Let's go for it cause location s couting comes up early this... https://t.co/U8FjMpDMSK

#valborg#eld #radiosporten #svt#sr #adidas#rök # @ Varb ergs fästning https://t.co/MPxQ7oVcJd

En helkväll som avslutades med ett biobesök på Bergakung en. Guardians of the galaxy är en skön... https://t.co/SqIncL 20S5

Det här är nog ändå favoriten. #wizard #diablo3 @ Godväde rsgatan https://t.co/xpyKpQQhX3

Vackert med #Valborg på flotte och körsång med @ronning eshow😊✨✨🎆🍀 @... https://t.co/9ijZxq25ma

Figure 5.2.: Screenshot of the plain list component filled with case study data.

The category list was also precisely implemented as the design suggested. It takes as input a title, the connected field, the filtered collection, the applied filter, and the color mapping. The filtered collection is ordered by count and rendered as a list with checkboxes corresponding to the filter state of the item. The included bar chart is maxed at the first item and displays the absolute count of the item. Upon filters or collection changes the list automatically re-renders. If checkboxes are changed, the new filter list is calculated and propagated to the core, which in turn applies it to all other collections. A screenshot of the component can be found in Figure 5.3.

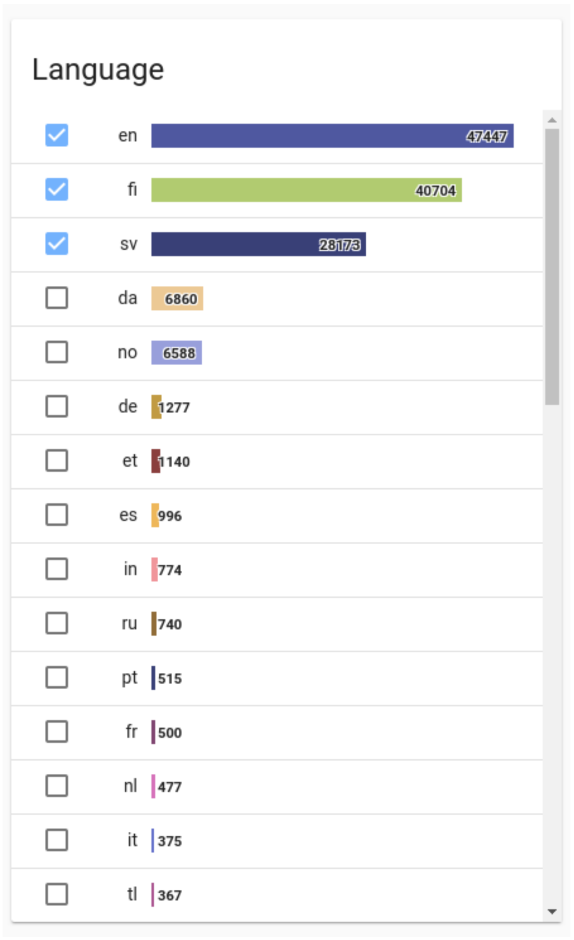


Figure 5.3.: Screenshot of the category list component filled with case study data and applied filters for *en*, *fi*, and *sv*.

5.2.2. Map Visualization

The map component was implemented closely to the design, but was extended with two minor improvements. The visualization was built upon the visualization library D3 and the topoJSON shapefile handler. The component takes not only the filtered collection and the targeted field as a parameter, but also the source of the topoJSONs and the identifying property field in the shapefile. Once the shapefiles are loaded, the shapes are mapped to the collection. The different levels were achieved by merging shapes together until the desired layer is achieved. This allows for quick changes in layers without recomputing and rendering all shapes.

The color-coding of the shapes is achieved by mapping the counts of the rendered, visible shapes on a quantized, six-step color scale. A blue color scheme is used by default because it is differentiated clearly from all colors used in the schemes for categorical data. With every change in the collection, the color-coding is recalculated and reapplied without a need to rerender the shapes. Over the visualization, a legend was added that displays the color-coding intervals and highlights the corresponding shapes upon hovering, thereby making the colors easier distinguishable.

The controls of the visualization were kept to a minimum. Every displayed shape offers a tooltip on hovering, providing the exact name of the shape and its parents (i.e., Sweden - Kronoberg - Växjö) together with the absolute amount of events for this shape. Once clicked, every shape displays a context menu offering to add or remove the corresponding region to the filter. Additionally, the context menu provides the possibilities to split a region up in its subregions and merge all regions of its parent region together, if the map material provides the needed levels. This allows the user to split specific regions into subregions while keeping others untouched and vice versa. Regions that are included in the current filter are recognizable by the shape having a red stroke around it. As an added feature, a button is floating in the upper right corner that resets the pan and zoom to the starting position. A screenshot of the menu controls can be found in Figure 5.4 while the whole component can be seen in Figure 5.5.

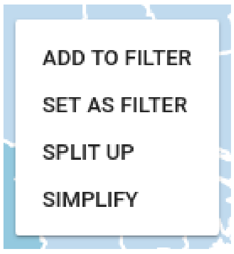


Figure 5.4.: Screenshot of the menu controls in the map component.

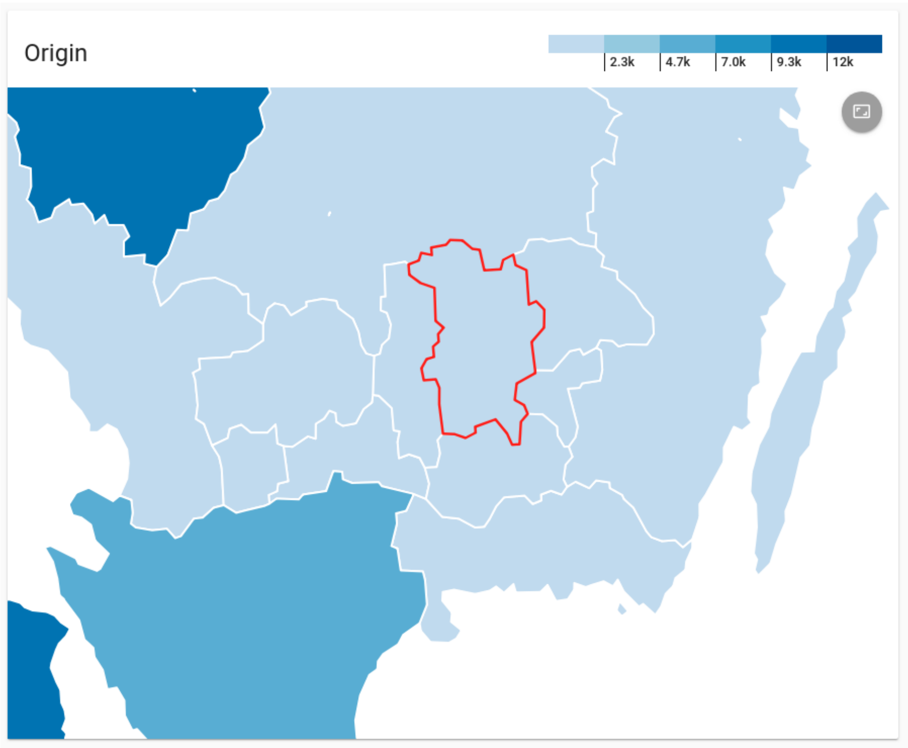


Figure 5.5.: Screenshot of the map component filled with case study data and filtered for Växjö. Zoomed in to show Kronoberg split up into municipalities while keeping the surrounding counties untouched.

5.2.3. Time Series Visualization

The time series component was also built with D3 and was set up to take not only the component title, the filtered collection, and the filters together with the displayed dimensions as parameters, but also requires the color-coding information for the category dimension and the extent of the time dimension. The visualization encapsulated its logic into the three parts outlined by the conception: (i) the *overview time series*, (ii) the *detailed time series*, and (iii) the control layer connecting the two time series.

Upon the initial rendering, the visualization sets up the three parts and binds event handlers. These are reported back to the containing component to enable triggers from outside. This way, changes in the collection or changed filters do not require a full re-render of the visualization, but can be displayed by transitioning the already rendered parts. To generate the stacked time series, the collection is stacked by the parser. For that, the parser first builds a data structure spanning the whole timeframe that is provided by the extent parameter. This way missing days in the collection are automatically filled. After that, the different groups are summed up, descendingly ordered by total counted, and then their stacked values for each timeslot are calculated by summing up all smaller groups count in that timeslot.

The **overview time series** was implemented in a straightforward manner. The stacked groups are rendered in their corresponding color in the highest provided granularity. To prevent a changing extent on the x-axis when changing filters, and thereby possibly removing outer timeslots, the extent of the axis is set to the supplied parameter.

The **detailed time series** is a more complex adaptation of the overview time series. The extent is set by the user selection in the overview time series. To provide a more meaningful view of the displayed data, the granularity is dynamically calculated. To achieve this, the physical width of the visualization is split to achieve a tick approximately every 50 pixels. In every tick, the nearest timeslots are aggregated by displaying the mean over all included values for every group. The lenses that are added by the *Focus+Context* approach split this time series up into multiple strung-together time series, applying the same principle for granularity. The values are split up into the selected domains and calculated and rendered independently. This leaves the problem that where the time series connect the values differ and there is a visible jump in the visualization. To overcome this problem, the outermost ticks were not displayed as the means of all their values, but as the exact values at the specific timeslot.

The **control layer** binds the two time series together and provides all the controls to the user. To select the detailed domain, the user can drag a so-called brush over the overview time series. A brush consists of two draggable handles. When dragging a handle, only the corresponding side of the extent changes, and when dragging the transparent area between the two handles, both handles and extent move in parallel. To hint the user at this functionality, the handles over the overview time series were connected to the edges of the detailed time series by dashed lines. To provide the *Focus+Context* functionality, two buttons were added right under the detailed time series. One button adding a lens when enough space is left and one button removing the innermost lens. The lens' controls extend the detail selection by providing a brush over the overview time series to control the domain of the length (*from* which date *to* which date) and a brush right under the detailed time series to control the range of the lens (*how much space* the lens domain shall take up). A dashed line also connects the two brushes, and the brushes are limited in their movement by their adjacent brushes. For every further lens, this approach is repeated. When moving a brush over the overview time series, the exact timeslots of this brush are displayed underneath the brush. The innermost domain, being a lens or just the detail selection, represents the filter that is applied to all other components. This is clarified by changing the brush and dashed line color and always displaying the selected timeframe in the upper right corner of the component. A screenshot of the component can be found in Figure 5.6.

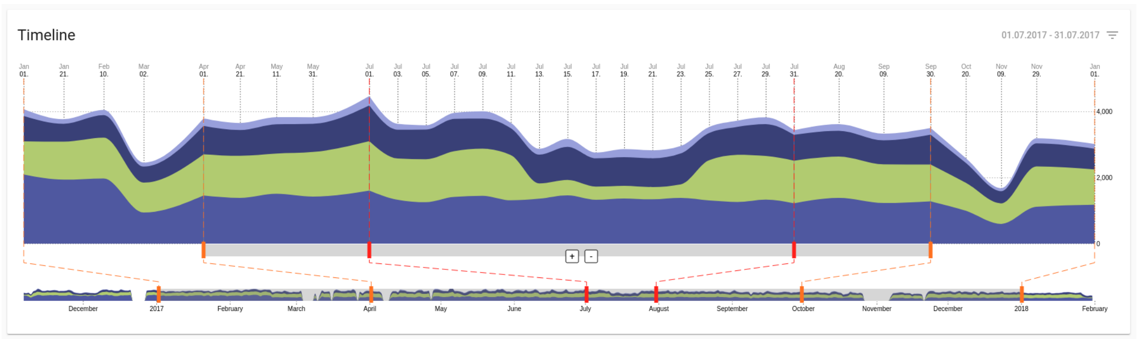


Figure 5.6.: Screenshot of the time series component filled with case study data and filtered for July 2017, showing four stacked languages and two lenses focusing in with the whole year 2017 as context.

5.3. Connectors

The concept of the connectors was designed to be as easy to implement as possible, enabling researchers to add new data sources and viewing angles quickly. Therefore, the only requirement necessary for connectors was to provide an HTTP *API* endpoint that queries the data source and returns results based on a provided filter, requested fields, and an optional limit. In the following sections, the work that was necessary to implement the *Twitter Language Usage* and the *Södra Forestry Yields* case study connectors is described.

5.3.1. Twitter Language Usage

The map visualization was built using the visualization library D3 and the shapefile handler TopoJSON. At first, the dataset had to be prepared for usage in the connector. The *NTS* dataset came in separate JSON files totaling 9.7 million Tweets and a size of 12.4 GB. The Tweets had to be parsed, filtered for sparse entries, and only the relevant fields picked. This boiled the dataset down to 2 million entries and a size of 413 MB. The steep reduction of Tweets came from the fact that most Tweets did not provide coordinates, as this is an opt-in feature, but is necessary to visualize the entries on the map. The picked fields were the ID, the timestamp, used language, coordinates, and the text of the Tweet, which reduced the size of the data considerably.

In the next step, the entries had to be enriched. For this use case, this meant to map the coordinates to a geographical region. The already introduced HASC was chosen, and map material searched that provided shapes for the Nordic countries down to the lowest possible level and included their HASC codes. Some research turned out sufficient map material that mapped Norway, Sweden, and Denmark down to municipalities [13]. For Finland, only maps on the county level could be found as Finnish subdivisions changed over half a dozen times this decade alone [14]. Iceland was included in the dataset but was not included in the map materials as the subdivision structure differs significantly from other countries. The dataset entries were mapped directly to the lowest possible level in the selected map material, and the HASC code was appended. Entries that could not be matched were discarded, which brought the total amount down to 1.7 million entries while not noticeably changing the size, due to the added field equalizing the missing entries.

These steps resulted in a dataset consisting of 1.7 million Tweets stored as JSON arrays in JSON files split for each day. This situation resembled the provided raw dataset very

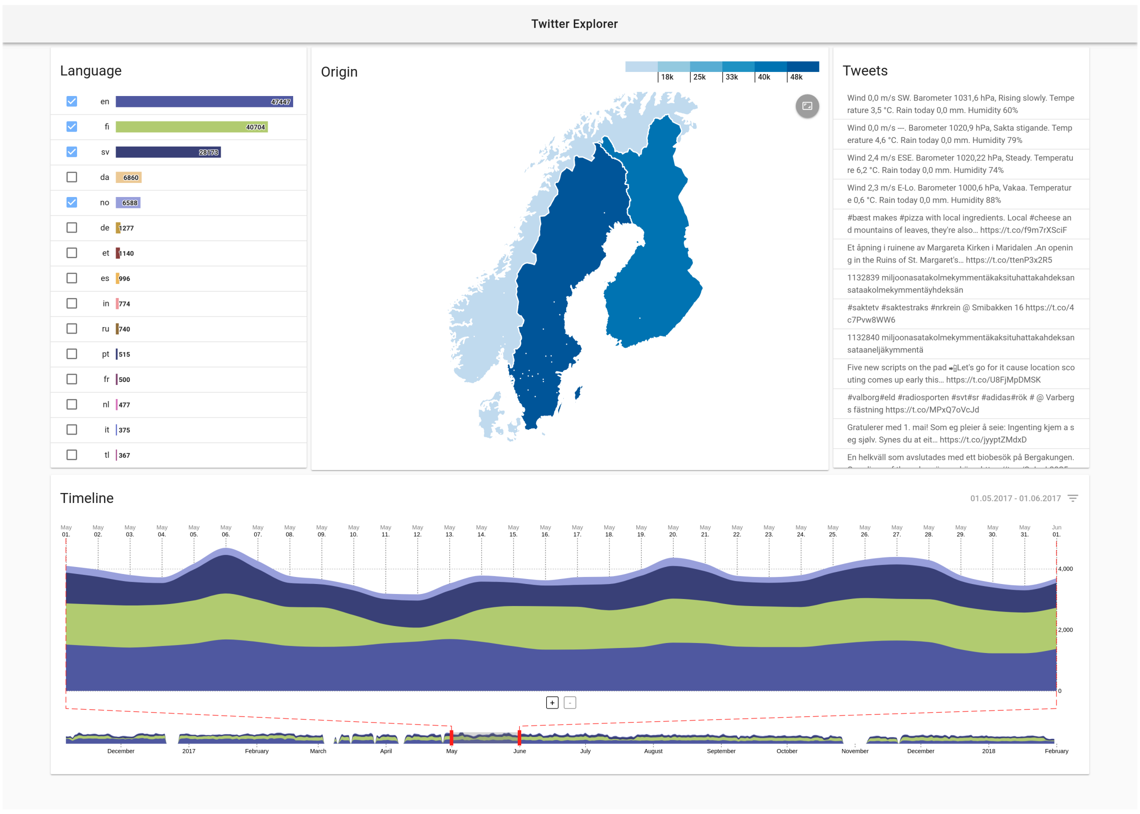
[13]https://github.com/deldersveld/topojson [Accessed: 06-Mai-2018]
[14]http://www.statoids.com/yfi.html [Accessed: 06-Mai-2018]

closely, and was not optimized with a real database to keep the use case simple and highlight what the system is capable of achieving even without an optimized data source.

The implementation of the connector was also kept as simple as possible and resulted in a *node.js* script adapting the filter implementation of the frontend to work on the JSON files and provide the server endpoint via *express*. Additionally, the connector provided an endpoint that takes the *URL* of a backend service and takes care of registering itself at the provided backend service. As the most common request scenario is the default filter set that is defined in the configuration, the connector was built to precalculate and cache the answer to this query, which greatly improved initial loading time for the frontend. The whole connector consisted of less than 100 lines of code, and the performance was analyzed in Chapter 6 *Evaluation*.

The configuration that was registered to the backend service defines the *URL* on the localhost, since the connector is deployed to the same machine as the backend service and defines the fields *date* (type: time), *language* (type: categorical) and *hasc2* (type: categorical). Additionally, the starting filters were defined with a preset for language and date. For the frontend, the configuration defined a category list for the field *language*, a map for the field *hasc2* with the selected maps, and a plain list for the field *text* in the first row. The second row was defined to only include a time series with the field *language* over the field *date* filling the whole row. The full configuration can be found in Appendix A.

The complete resulting frontend is displayed in Figure 5.7 and was used for the experiment and the expert interviews described in Chapter 6. A working instance of the system with the presentation of this case study can be found online [15].



Language

✓	en	47447
✓	fi	40704
✓	sv	28173
☐	da	6860
✓	no	6588
☐	de	1277
☐	et	1140
☐	es	996
☐	in	774
☐	ru	740
☐	pt	515
☐	fr	500
☐	nl	477
☐	it	375
☐	tl	367

Tweets

Wind 0,0 m/s SW. Barometer 1031,6 hPa, Rising slowly. Temperature 3,5 °C. Rain today 0,0 mm. Humidity 60%

Wind 0,0 m/s ---. Barometer 1020,9 hPa, Sakta stigande. Temperature 4,6 °C. Rain today 0,0 mm. Humidity 79%

Wind 2,4 m/s ESE. Barometer 1020,22 hPa, Steady. Temperature 6,2 °C. Rain today 0,0 mm. Humidity 74%

Wind 2,3 m/s E-Lo. Barometer 1000,6 hPa, Vakaa. Temperature 0,6 °C. Rain today 0,0 mm. Humidity 88%

#bæst makes #pizza with local ingredients. Local #cheese and mountains of leaves, they're also… https://t.co/f9m7rXSciF

Et åpning i ruinene av Margareta Kirken i Maridalen .An opening in the Ruins of St. Margaret's… https://t.co/ttenP3x2R5

1132839 miljoonasatakolmekymmentäkaksituhattakahdeksansataakolmekymmentäyhdeksän

#saktetv #saktestraks #nrkrein @ Smibakken 16 https://t.co/4c7Pvw8WW6

1132840 miljoonasatakolmekymmentäkaksituhattakahdeksansataaneljäkymmentä

Five new scripts on the pad Let's go for it cause location scouting comes up early this… https://t.co/U8FjMpDMSK

#valborg#eld #radiosporten #svt#sr #adidas#rök # @ Varbergs fästning https://t.co/MPxQ7oVcJd

Gratulerer med 1. mai! Som eg pleier å seie: Ingenting kjem av seg sjølv. Synes du at eit… https://t.co/jyyptZMdxD

En helkväll som avslutades med ett biobesök på Bergakungen.

Figure 5.7.: Screenshot of the complete frontend with the configuration and dataset of the *Twitter Language Usage* case study.

[15] https://twitter.multiscale-explorer.de/

5.3.2. Södra Forestry Yields

The Södra dataset came in the form of a 29 MB comma separated values file (CSV-file) and contained over 345.000 entries. Every entry represented a cut-down tree and consisted of a timestamp, three-dimensional coordinates, some measurements of the tree, and the species name. The dataset did not show any sparse entries or inconsistencies. To make the dataset ready for exploration, only the coordinates needed to be mapped to a map of Sweden as already described in the previous case study. Finally, the enriched dataset was saved back to a CSV-file.

The connector was implemented as lightweight as possible and loads the data source into memory at startup for better performance. Due to the small size of the dataset, this resulted in under 20 MB of additional RAM usage by the connector. Upon queries, the connector iterates over the whole dataset in memory and applies the queried filters. The whole connector was written in *node.js* and copies the filter implementation of the frontend.

As for the configuration, the case study featured a category list for the species, a map for the coordinates, and a time series displaying the species over time. Additionally, a further category list was added displaying the number of logs generated from a tree by treating the numerical dimension as a categorical one. For future presentations, additional components for numerical dimensions could be added, such as, for example, a histogram of scatter plots. The complete resulting frontend is displayed in Figure 5.8 and the configuration can be found in Appendix B.

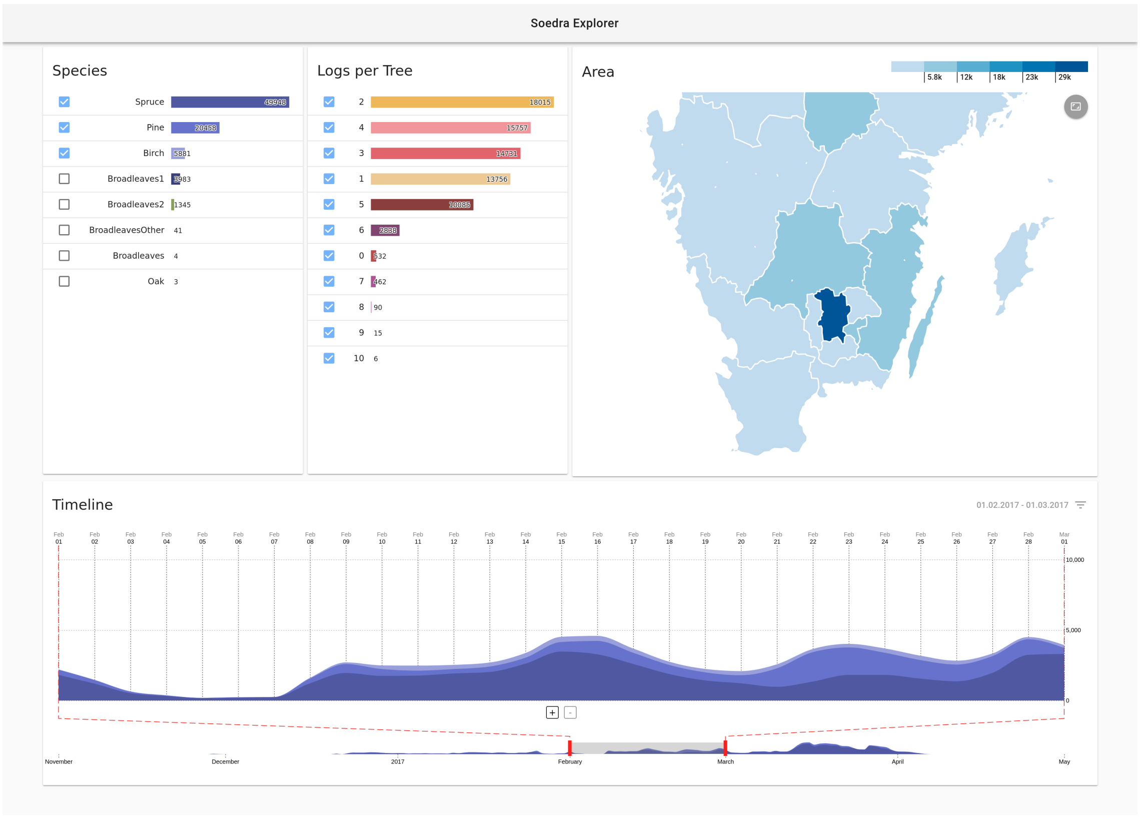


Figure 5.8.: Screenshot of the complete frontend with the configuration and dataset of the *Södra Forestry Yields* case study.

6. Evaluation

The evaluation phase was the final phase of the project. In this chapter, first, the design of the experiment and the expert interview are described. After that, the findings of the experiment, the questionnaire, and the interviews are reported and explained. Lastly, they are interpreted and put into the context of the requirements.

6.1. Design

This section describes the setup of the experiment and the expert interview sessions to make them reproducible and help to make the concluding findings intelligible.

6.1.1. Experimental Setup

The goal of the experiment was to capture reproducible, quantitative metrics of the performance of the system under a realistic deployment situation and expected load and tasks. As the performance of such a data-intensive distributed system relies heavily on the many factors of the used environment, these variables have to be locked down and documented to make sure the experiments with the system are reproducible.

On the client-side, the target was to simulate a consumer laptop. The experiment setup was running an up-to-date Ubuntu 16.10 LTS operating system and featured a single CPU core of an Intel(R) Core(TM) i7-6500U at 2.50GHz. The system also included a GM108GLM Quadro K620M GPU and 8 GB of DDR3 RAM. The amount of RAM was high enough that no swap needed to be used, leaving the HDD as a non-factor. The browser Chromium 66 was used which comes bundled with v8 6.6. The connection between server and client was throttled to a symmetric 10mbps connection. A full specification list for the client setup can be found in Table 6.1.

OS	Ubuntu 16.10 LTS
CPU	single 500MHz core
GPU	NVIDIA Quadro K620M
Runtime	Chromium 66.0.3359.139
JS engine	v8 6.6.346.26
Connection	10mbps symmetric

Table 6.1.: The specifications of the client used in the experiment.

On the server-side, an expectable virtual machine instance on a data center cluster was remodeled. It was also running an up-to-date Ubuntu 16.10 LTS operating system and was housing both the connector and the backend service, as intended by design. Hardware-wise, the experimental setup included two cores of an Intel(R) Xeon(R) CPU E3-1245 V2 @ 3.40GHz, 32 GB DDR3 RAM and a SSD capable of a mean sequential read speed of around 500M/B. Again, the RAM capacity was large enough for both services and no swap usage was needed. As a runtime environment for the services, the current node.js LTS version was used, which packs node v8 6.0. A full list of all specifications of the server environment used in the system can be found in Table 6.2.

OS	Ubuntu 16.10
CPU	2x cores @ 3.40GHz
SSD	ca 500M/B sequential read
Runtime	node.js v8.4.0
JS engine	v8 6.0.286.52

Table 6.2.: The specifications of the server used in the experiment.

The experiment consisted of two parts: (i) a basic performance testing for an instance of StanceXplore, and (ii) an in-depth performance testing of the first case study. The performance test of StanceXplore consisted of measuring the resources needed to load the frontend, memory usage, and the necessary scripting time to render the presentation. The dataset used had the same source as the case study but only included Tweets from May 2016 in Sweden. The restriction on the collected metrics was because replicating StanceXplore in the same environment as the case study and filling it with the same dataset was not feasible. The metrics were supposed to give a rough comparison to the new system while keeping the differences in functionality in mind.

The performance measurements on the *Twitter Language Usage* case study included more metrics and was conducted on different samples of the dataset. All measurements were taken for different sample sizes of the dataset defined by the number of entries: 100k, 500k, 1m, and 1.5m samples were selected. For every sample size, there were 100 iterations conducted with each shuffling the dataset and randomly picking its entries before injecting the subset into the connector.

The first set of metrics was the **resources** needed to be loaded by the frontend. This included the dataset with the defined amount of entries and the resulting size, plus the loaded dependencies. These dependencies include the HTML, CSS, and JS files necessary for the presentation. They can be categorized in one-time downloads, which are subsequently loaded from cache, and recurring downloads.

The second set of metrics captured were centered on **timing**. The loading times of the configuration and the topoJSONs for the map component were measured as dataset-size-independent metrics. In all cases, the loading times include not only the theoretical transmission time, but also the connection overhead, the lookup time on the server, and the browser internal parsing to an object. Depending on the size and sample of the dataset, the Time-to-Interactive (TTI), the loading, and the parsing of the dataset itself were also measured. TTI describes the time a website needs from the initial call until the user can make the first meaningful interaction. In the case of the frontend, this means until all visualizations and components are rendered, and the loading state is removed.

Additionally, the **memory usage** of the presentation was measured. This was done after the initial dataset has been loaded and the presentation was waiting for user interactions. At this point, the dataset was already parsed, filtered, and the filtered collections passed and rendered in all components. It is important to measure this metric as the RAM usage takes a toll on power usage and, in combination with other tools running on the client environment, clock up the system, forcing the use of swap storage and thereby slowing the whole environment down.

Lastly, the time needed to interact with the system depending on the size of the dataset had to be measured. To achieve this, three different **filtering times** were captured. These results would rely heavily on the picked sample because, for example, whole categories could be missing from a dimension. As in every iteration the sample entries were chosen randomly, possible outliers were balanced out. The first measurement was the addition of

Sweden to the map filter. The second measurement was filtering the time series for the month July 2017. And lastly, the toggling of the largest category in the language category list was measured. All these filter changes were made independently of each other and on the preset default filters. An overview of the collected metrics for the case study and the process can be found in Figure 6.1.

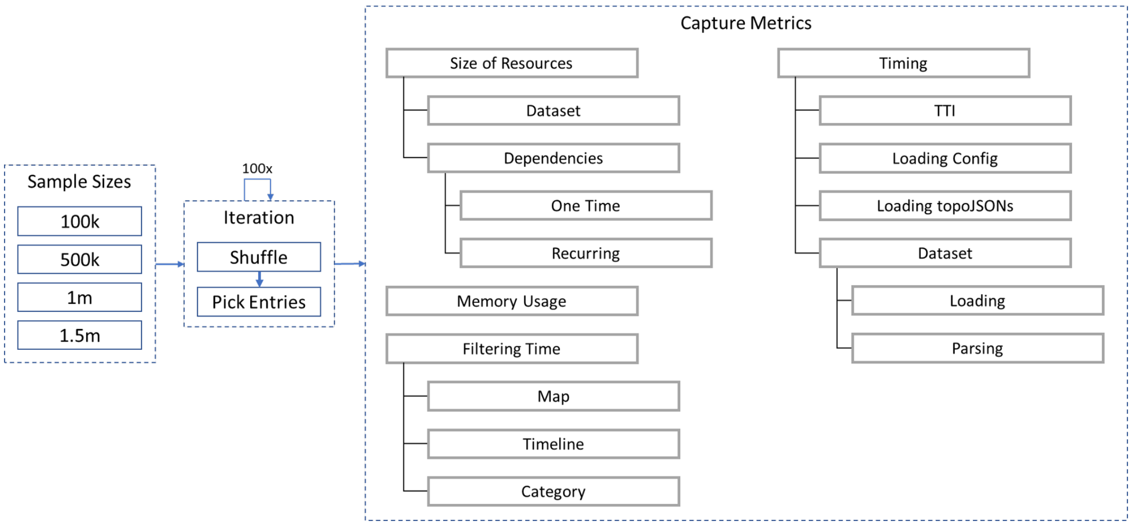


Figure 6.1.: Overview of the collection process and the collected metrics for the experiment.

6.1.2. Expert Interview

The expert interview consisted of five parts and was conducted with five experts in the field of data visualization and data exploration. All experts were male and ranged from Ph.D. students to senior researchers and lecturers from the Department of Computer Science and Media Technology of Linnaeus University. The selection and invitation were organized by the supervisors of this thesis project to ensure a level of independence.

At first, the interviewee was given an **introduction** to the system. The creation of the tool was shortly motivated, and the case study data and scope was explained. A quick privacy notice was given, and the interviewee was asked if he consents to a temporary recording of the following conversations to record all findings. At this point, the functionalities of the system were explained and presented. All of these steps were performed in-person by the interviewer with the help of the demonstration system. Next, the interviewee was given five minutes of **trial** time to play around with the tool to get a grip and feel comfortable utilizing the different components. After that, the interviewee was asked to answer five questions with the tool in a **think-aloud session**. These questions were designed to reveal if the interviewee understood the concept behind the connected filters and the multiscale visualizations. The full set of questions is listed in Table 6.3.

Q1 was designed to set the basis for the following questions. The interviewee had to adjust the detail timeframe on the time series view and navigate through the map to reach the municipality of Stockholm. The answer could then be found in the tooltip of the shape. **Q2** checked the understanding of the connected filters and the handling of the category list. The interviewee had to set Stockholm as a filter in the map visualization and then simply read the displayed number of the Swedish entry in the category list. **Q3** was designed to again check the understanding of the category list component. The answer could be given by simply comparing the displayed bars visualizing the amount of Tweets without any

Q1	*How many Tweets were created in 2018 in the municipality of Stockholm for the preset languages?*
Q2	*How many of those were in Swedish?*
Q3	*How does this roughly compare to Tweets in other languages?*
Q4	*How does this roughly compare to the surrounding municipalities?*
Q5	*How does the month July 2017 roughly compare to the rest of the year in the municipality of Stockholm for the languages English and Swedish?*

Table 6.3.: List of questions asked in the think-aloud session.

interactions with the tool. **Q4** had the goal of checking if the interviewee could apply the same thought process to the map component by reading the color-coding of the surrounding municipalities without any interactions. The last question (**Q5**) stood alone as it did not build upon the setup provided by the first question and had the purpose of providing insights into the handling of the multiscale aspects in the time series visualization. To answer the question the interviewee had to set the language filters, then set the detail view of the time series to include the whole year 2017, and add a lens providing a focus on the month July while keeping the context of the rest of the year.

After answering all questions, the interviewee was asked to fill out the **UEQ**. The *UEQ* itself and the idea behind it is described in Section 3.2.

The last part of the interview sessions was the actual **interview**. They were conducted as open interviews with the main purpose of getting feedback on the used concepts, the usability and understandability, and generic improvement possibilities. Directions the interviewee was asked about were the concepts of the connected filters, multiscale in the time series visualization, and multiscale in the map visualization.

6.2. Findings

In this section, the findings of the measurements taken in the experiment, the results of the survey and the findings in the expert interview are laid out. Any interpretations and implications for the requirements are deferred to the next section.

6.2.1. Performance Measurements

The measurements done for StanceXplore showed that the selected dataset contained over 118.000 entries. The transmitted dataset took up 21.3 MB, as every entry was included as a separate object with a selection of fields. Additionally, 800 KB of additional resources were loaded which would be cached for further visits. This brings the total required resources for the presentation up to 22.1 MB. The presentation and, specially, the visualizations, took up 2,168 ms of scripting time resulting in 85.3 MB of memory usage.

The measurements done for the case study can be separated into two categories: (i) the static metrics, and (ii) the sample dependent category.

The static category consists of metrics that stay the same independently from the sampled dataset. Nevertheless, all metrics were taken with every sample size and every iteration. The static metrics included the loading time of the configuration and the topoJSONs and the size of the dependencies. The configuration is only 756 bytes large and averaged a loading time of 99 ms. The one-time dependencies consisted of generic dependencies and fonts. Generic dependencies are HTML, CSS, and JS files while font files are only loaded if the client environment does not already provide the requested font. Generic dependencies came in at 250.9 KB and all fonts at 93.7 KB, bringing the one-time dependencies to a total of 344.6 KB. The only recurring dependencies are the topoJSONs of the map component. These were considered recurring as they are provided by the configuration and could be hosted externally, leaving the caching policy up to the hosting instance. In the instance of this case study, the map material for Denmark, Finland, Norway, and Sweden sum up to a total of 124.5 KB, taking 231 ms as the system downloads them in parallel. The full list of findings from the sample-dependent metrics are reported in Table 6.4, and the analysis of these findings can be found in Section 6.3 *Results*.

sample size	100,000	500,000	1,000,000	1,500,000
cross products	47,422	139,314	212,496	267,957
dataset size (KB)	626	1,762	2,750	3,501
dataset loading (ms)	446	692	892	1,075
dataset parsing (ms)	81	162	328	438
TTI (ms)	3,311	4,675	5,862	6,247
filter map (ms)	123	301	451	527
filter category (ms)	170	293	423	457
filter time series (ms)	38	72	98	126
memory (MB)	56.3	68.4	83.3	93.1

Table 6.4.: Findings of the sample-dependent metrics captured in the case study.

6.2.2. User Experience Questionnaire (UEQ)

The *UEQ* groups its six scales into three groups: *hedonic qualities*, *pragmatic qualities*, and *attractiveness*. Both for the scales as for the groups, values between -3 and +3 are calculated. Values between -3 and -0.8 represent negative evaluation, values between -0.8 and +0.8 evaluate to neutral, and values larger then +0.8 evaluate positively. Due to the avoidance of extreme values, results outside of a -2 to +2 range are unlikely. The questionnaire provides an extensive comparison dataset which contains mostly established systems and only few novel or experimental systems. Expected problems with the results were discussed in Section 3.3. The mean values per scale were visualized in Figure 6.2.

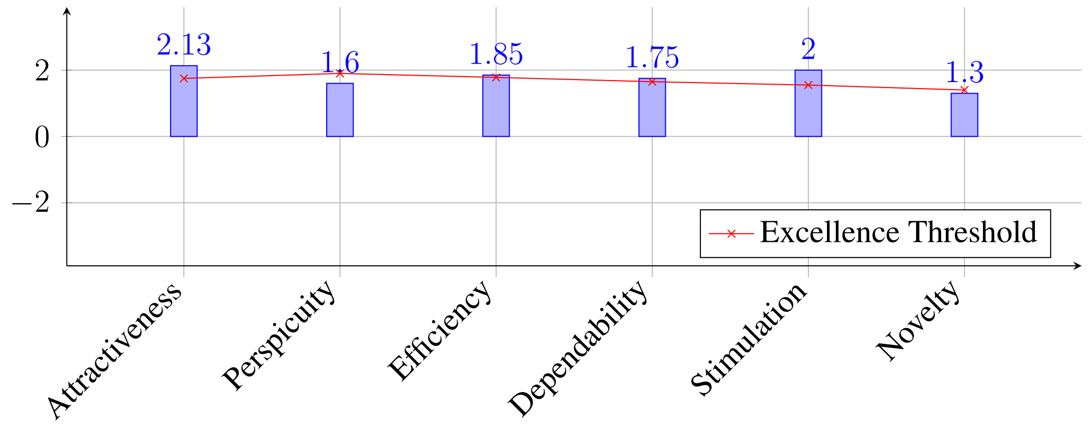


Figure 6.2.: User Experience Questionnaire, mean results per scale.

Overall, all scales and groups scored positively, with *novelty* being the least performing scale with a score of 1.3. In the scale of *attractiveness*, a mean of 2.13 was reached.

Compared to the reference dataset, this corresponds to an *excellent* score, leaving the tool in the range of the 10% best results.

The pragmatic qualities of the system scored with a mean of 1.73. *Efficiency* reached a mean of 1.85, putting it in the top 10% range. *Perspicuity* had a mean of 1.6, classifying it in comparison as *good*, which can be interpreted as lower than the 10% best results but better than 75% of the comparison results. *Dependability* reached 1.75, putting it again in the *excellent* field.

The hedonic qualities scored 1.65 overall, making it the least performing group. *Novelty* had a mean of 1.3, categorizing it as *good*. *Stimulation* also reached the *excellent* field with a mean of 2.0.

6.2.3. Expert Interview

The expert interviews started with a trial after a short introduction. As the tasks were explicitly designed to reveal if the user understood the concept behind the connected filters and the multiscale visualizations, it became clear that all interviewees understood both concepts and had no problem using the system. All tasks were solved correctly without any meaningful hurdles, although different approaches to get to the answers were used. Both in the trial and in the interview part the interviewees stated that they did not encounter any conscious delay in the system or experienced generic context disruptions.

The multiscale time series component was perceived as complex. Not all interviewees utilized the full potential of the applied multiscale techniques, but instead used more profound approaches to solve the tasks. Two users used the *Overview+Detail* part of the visualization but did not work with the lenses of the *Focus+Context* technique as they felt that the tasks were also solvable by using temporal differentiation: First memorizing the first timeframe and then comparing that to the second. These interviewees stated that the *Focus+Context* technique was not necessary, but could see that there would be specific tasks where this technique would be of benefit. Overall all interviewees agreed on the benefit of this multiscale visualization over a simple time series.

The map component was more clearly understood and did not pose a hurdle. The users were swift to interact with the visualization and liked the possibilities provided by the multiscale approach. As feedback, it was suggested to add an initial filter on the map to make it more clear what is filtered and what is not.

Overall the users wished for better information on hover. The tooltips were perceived as too small and with too little context in both large visualizations. The added benefit of the system was visible for all interviewees, and they felt confident in their usage of the system. Asked if they wished for an undo button to make their actions easier to reproduce, every interviewee declined.

6.3. Results

In **comparison** with *StanceXplore*, the system mainly improved on adaptability and performance, since *StanceXplore* was not designed with these as priorities. Featurewise, the system added functionality in the frontend regarding interactivity and multiscale capacity in both the map and time series, while only dropping features such as the hashtag grid which were out of scope for this project.

Regarding the performance metrics, the experiment revealed that the compacting algorithm provides a clear benefit. While StanceXplore needed to transmit over 20 MB of data for around 100 thousand entries, the newly developed system only required a mean of 626

KB in the comparable case study — a reduction of 97%. With 800 KB for *StanceXplore* and 345 KB for the new system, the dependencies are no problem in both cases. With a memory usage of 85.3 MB, *StanceXplore* is surprisingly good for the size of the dataset, but compared with the new system it uses 52% more memory for the same amount of entries. Concluding these findings, the new system outperforms *StanceXplore* for the studied use case.

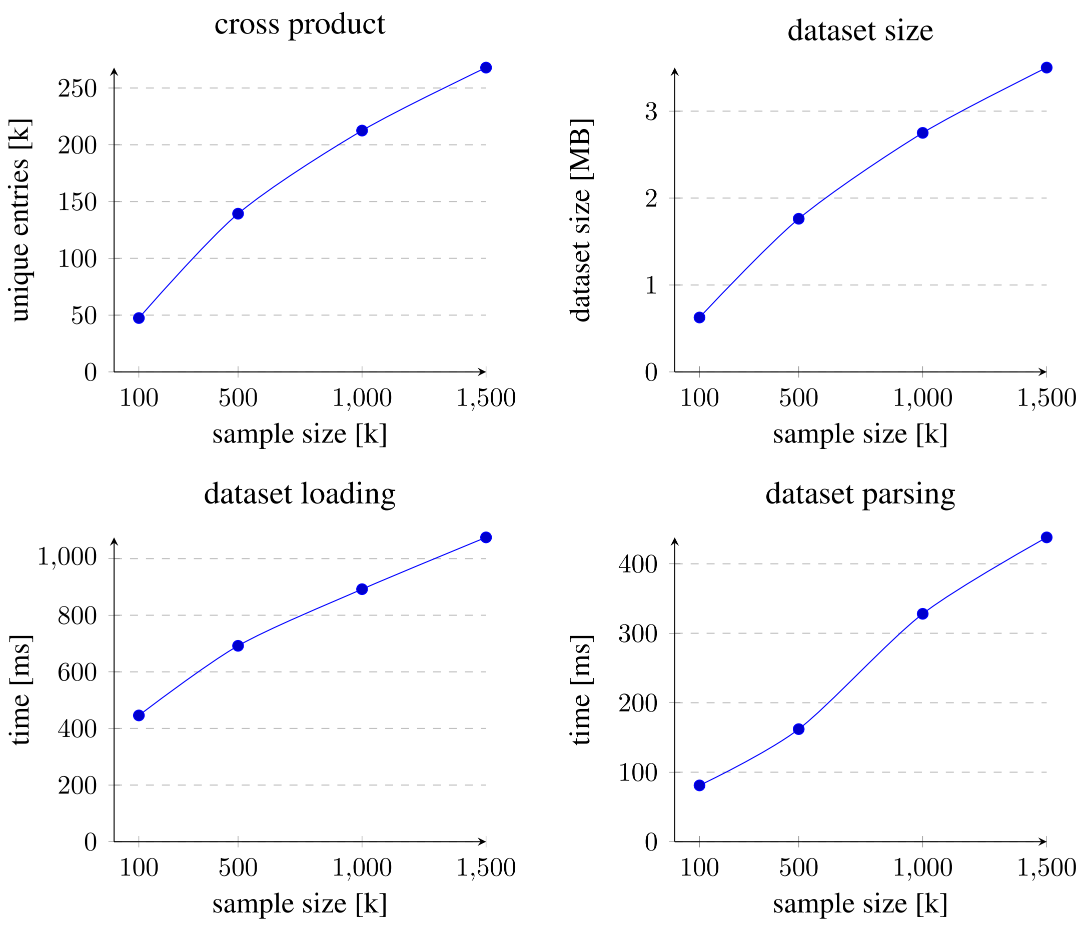


Figure 6.3.: Finding for the metrics directly connected to dataset handling.

By looking at the findings of the sample-dependent **metrics**, conclusions about scalability could be drawn. As the compacting algorithm takes all unique values in a dimension and later aggregates entries into their dimension combination, data points are more likely to be already fitting into an existing combination the more data points are already aggregated. This should result in a smaller than linear growth of the cross product. As seen in Figure 6.3, this is indeed the case, with all other metrics tied to the cross product following the same trend. Parsing the transmitted dataset showed a jump between 500.000 and a million data points in the sample which later could be confirmed in a smaller experiment that was conducted with more sample sizes. It seems like this performance loss is originating from hitting the CPU limit, but no conclusive reason could be found.

Looking at the interaction delays represented in the experiment by the filtering times, it became clear that for all chosen sample sizes the delay should not be noticeable. As seen in Figure 6.4, only the map filtering broke the half-a-second mark with the largest sample size. For interactions on the map and category list component, the needed time is growing slower than linear, which could be explained by the same reason that also

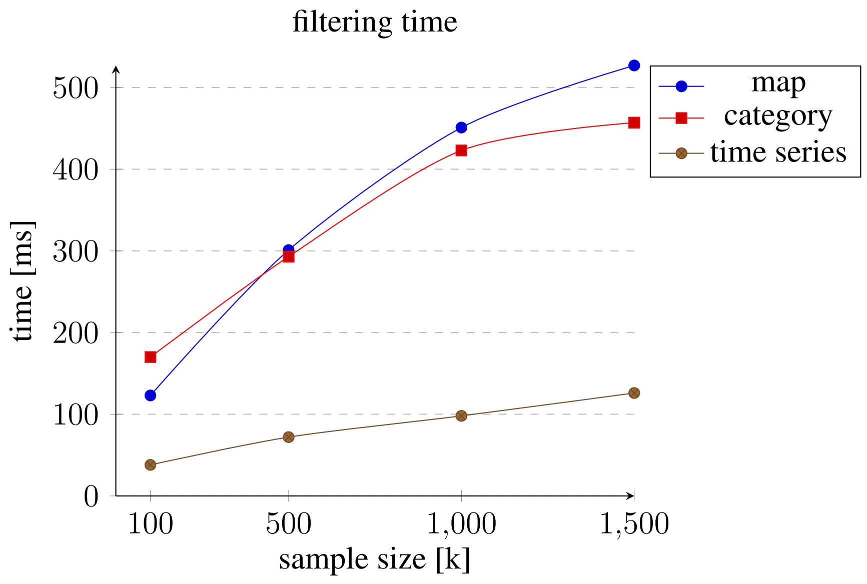


Figure 6.4.: Findings for the filtering time metrics.

explains the substantially smaller linear filtering time for the time series component: The stacking algorithm that is required to visualize the stacked area chart is comparably slow and mainly growing based on the extent of the time dimension. Stacking many layers is not as calculation-heavy as filling all gaps in the time span. As filtering in the time series component does not trigger a refiltering and rerender of the same component, this action is considerably faster than all others.

The **UEQ** showed clear strengths in pragmatic qualities and attractiveness, while overall turning out a satisfying image. These findings should be taken as preliminary results due to the small number of participants.

With all findings concluded, the **requirements** needed to be checked. The accessibility requirements (Requirement 1) were met in the conception phase with the design decision to use a distributed architecture with the browser as the client for the enduser. The adaptability requirements of Requirement 2 were the main driving points behind the component concept of the architecture and the additional implementation of the second case study proved the fulfillment of all sub-requirements. Requirement 3 sets full filterability as a requirement, which was met in the conception phase by the combined filtering concept linking all components together. The multiscale requirement of Req. 4 expected a multiscale map and multiscale time series component, which were subsequently designed and implemented. The usability requirements of Req. 5 were more numerous and partly had to be checked by the expert interview. Requirement 5.1 and Req. 5.2 were taken into account in both conception and implementation phase and were confirmed in the trial and task phase of the expert interviews. Requirement 5.3 was met with the design and implementation of the compaction and compression algorithm, which was also tested and confirmed in the experiment. Requirement 5.4 required a clear to understand and differentiate usage of color schemes which was taken into account in the implementation, but turned out to be non-optimal as the repetition of colors in large categorical dimensions could lead to similar colors in the same proximity. This did not lead to confusion for the interviewees but was noted by multiple interviewees. Requirement 5.5 to Req. 5.9 were tested in the expert interview and either confirmed by the interviewees or proven in the trial and task phase.

7. Conclusion

This chapter discusses the results of the project, draws conclusions from the findings, and analyzes if the designed system answered the initial research question. At the end of the chapter, an outlook is provided by looking into possible future work and further opportunities for the evolution and usage of the system.

7.1. Discussion

The project turned out a working system for effective and efficient exploration of large multi-dimensional datasets. Additionally, two new multiscale visualizations were conceived, implemented, and evaluated for a map and a time series view. The evaluation of the system showed that all set requirements were met and the practical usability was proven by two case studies. The performance of the system turned out to be sufficient and scaled well, but not only regarding the compacting algorithm small improvement possibilities came up during the project. All improvement possibilities and ideas were documented both in this report and the source code of the tool itself, so that they may be addressed in future iterations. From this standpoint, it was shown that the design conceived in the conception phase provided a sufficient answer to the research question.

In comparison to previous work, the system improved in many points but left some features behind that were out of scope for this project. In comparison with StanceXplore, the system featured significantly-improved performance and scalability. Additionally, the functionality of the presentation was enhanced, and the focus moved to support higher adaptability. This focus change resulted in small missing features that are highly specific like the hashtag grid StanceXplore provided. However, the component structure of the new system allows for easy additions of new components and the inclusion of such in new or existing presentations. The time series visualization was influenced by the MultiStream concept, but differs in significant points. While the MultiStream concept used steamgraphs to visualize time series, the new system utilized less-complex stacked area graphs. One of the reasons for this move was that the focus of this visualization was not on the visualization of hierarchical time series, but instead on providing a more complete interactive multiscale experience. This focus also lead to the absence of granularity changes in the categorical dimension. Instead, the system improved on the concept with a flexible granularity in the time dimension providing better scalability for vast time spans. Additionally, the system provides improved controls over the detailed domain and full control over flexible lenses providing a focus into the data. One final big step is the integration of the visualization into a full system which is moving the implementation away from a pure proof-of-concept like MultiStream.

7.2. Outlook

For the future, there are two topics covered in this project. The first one is the resulting system and its further development and usage, and the second one is the research done in the field of multiscale visualizations.

The system was designed to be used for initial data exploration by non-data scientists and easy adaption to new data sources and viewing angles. The implementation of the case studies and the evaluation showed that the system is capable of fulfilling this job. StanceXplore was developed to answer a very similar question as the first case study tried to answer. For further projects like this, no new system would need to be developed, but instead, only a connector would need to be set up. If particular visualizations would be required, these could be included in the frontend, and all further projects could also benefit from it. Regarding further development, there are many options for additional components that would add more benefit to the system. For example, handling of numerical dimensions with visualizations for histograms or scatterplots. Also, the steps taken by the first case study connector implementation could be improved: a caching of the already compacted and compressed default filter result could be held at the backend service, greatly improving the initial performance of presentations of static datasets.

As for the research into multiscale visualizations in this project, a few conceptual improvements and further research possibilities emerged. Firstly, the *Focus+Context* approach inside an *Overview+Detail* approach showed promising results as already shown in previous work. But the addition of dynamic focusing lenses went one step further and improved the usability and possibilities of the visualization. In the evaluation and during implementation, it became clear that further conceptual improvements could yield great benefits. For example, better handling of granularity changes and interaction with the lens handles. Secondly, the evaluation showed that some users did not use the functionalities provided by the *Focus+Context* approach, but instead preferred to build an image in their mind by comparing temporally-separated visualizations using a *zooming* approach. To determine a use case for the *Focus+Context* approach and differentiation from use cases for other similar approaches like the one used by these users would be interesting.

In summary, there are promising possibilities for both the system and the research done that hopefully will be picked up in the future.

Bibliography

[1] R. Spence, *Information visualization: An introduction*, 3rd ed. Springer Publishing Company, Inc., 2014.

[2] M. Friendly and D. J. Denis, *Milestones in the history of thematic cartography, statistical graphics, and data visualization: An illustrated chronology of innovations*. Statistical Consulting Service, York University, 2008.

[3] K. W. Brodlie, L. A. Carpenter, R. A. Earnshaw, J. R. Gallop, R. J. Hubbold, A. M. Mumford, C. D. Osland, and P. Quarendon, *Scientific visualization: Techniques and applications*. Springer Science & Business Media, 2012.

[4] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual analytics: Definition, process, and challenges," *Lecture Notes in Computer Science*, vol. 4950, pp. 154–176, 2008.

[5] A. Kerren and F. Schreiber, "Toward the role of interaction in visual analytics," in *Simulation Conference (WSC), Proceedings of the 2012 Winter*. IEEE, 2012, pp. 1–13.

[6] E. Cuenca, A. Sallaberry, F. Wang, and P. Poncelet, "Visualizing hierarchical time series with a focus+context approach," in *IEEE VIS–Posters*, 2017.

[7] A. Kerren, I. Jusufi, and J. Liu, "Multi-scale trend visualization of long-term temperature data sets," in *Proceedings of SIGRAD 2014, Visual Computing, June 12-13, 2014, Göteborg, Sweden*, no. 106. Linköping University Electronic Press, Linköpings universitet, 2014, pp. 91–94.

[8] A. Cockburn, A. Karlson, and B. B. Bederson, "A review of overview+detail, zooming, and focus+context interfaces," *ACM Computing Surveys*, vol. 41, no. 1, pp. 2:1–2:31, 2009.

[9] M. Hasan, F. F. Samavati, and C. Jacob, "Interactive multilevel focus+context visualization framework," *The Visual Computer*, vol. 32, no. 3, pp. 323–334, 2016.

[10] E. Hadjidemetriou, M. D. Grossberg, and S. K. Nayar, "Spatial information in multiresolution histograms," in *Computer Vision and Pattern Recognition. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[11] L. Heide, *Punched-card systems and the early information explosion, 1880–1945*. JHU Press, 2009.

[12] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler, "Visual analytics: Scope and challenges," in *Visual data mining*. Springer, 2008, pp. 76–90.

[13] E. Cuenca, A. Sallaberry, F. Y. Wang, and P. Poncelet, "MultiStream: A multiresolution streamgraph approach to explore hierarchical time series," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018.

[14] ISO, “ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs),” International Organization for Standardization, Geneva, Switzerland, Tech. Rep., 2000.

[15] W. Aigner, S. Miksch, H. Schumann, and C. Tominski, *Visualization of time-oriented data*. Springer Science & Business Media, 2011.

[16] R. L. Harris, *Information graphics: A comprehensive illustrated reference*. Oxford University Press, 2000.

[17] S. Havre, E. Hetzler, P. Whitney, and L. Nowell, “ThemeRiver: Visualizing thematic changes in large document collections,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 9–20, 2002.

[18] L. Byron and M. Wattenberg, “Stacked graphs–geometry & aesthetics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, 2008.

[19] N. Thrower, *Maps and Civilization: Cartography in Culture and Society, Third Edition*. University of Chicago Press, 2008.

[20] A. H. Robinson, *Early Thematic Mapping in the History of Cartography*. University of Chicago Press, 1982.

[21] J. K. Wright, *Notes on Statistical Mapping: With Special Reference to the Mapping of Population Phenomena*. American Geographical Society, 1938.

[22] B. D. Dent, J. S. Torguson, and T. W. Hodler, *Cartography: Thematic map design*. WCB/McGraw-Hill Boston, 1999, vol. 5.

[23] R. M. Smith, “Comparing traditional methods for selecting class intervals on choropleth maps,” *The Professional Geographer*, vol. 38, no. 1, pp. 62–67, 1986.

[24] C. A. Brewer, A. M. MacEachren, L. W. Pickle, and D. Herrmann, “Mapping mortality: Evaluating color schemes for choropleth maps,” *Annals of the Association of American Geographers*, vol. 87, no. 3, pp. 411–438, 1997.

[25] ISO, “ISO 3166: Codes for the representation of names of countries and their subdivisions,” International Organization for Standardization, Geneva, Switzerland, Tech. Rep., 2013.

[26] G. Law, *Administrative subdivisions of countries: a comprehensive world reference, 1900 through 1998*. McFarland, 1999.

[27] J. J. Van Wijk and W. A. Nuij, “Smooth and efficient zooming and panning,” in *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*. IEEE, 2003, pp. 15–23.

[28] H. Hauser, “Generalizing focus+context visualization,” in *Scientific visualization: The visual extraction of knowledge from data*. Springer, 2006, pp. 305–327.

[29] Y. K. Leung and M. D. Apperley, “A review and taxonomy of distortion-oriented presentation techniques,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 1, no. 2, pp. 126–160, 1994.

[30] G. W. Furnas, "The fisheye view: A new look at structured files," Citeseer, Tech. Rep., 1981.

[31] M. Sarkar and M. H. Brown, "Graphical fisheye views," *Communications of the ACM*, vol. 37, no. 12, pp. 73–83, 1994.

[32] G. W. Furnas, *Generalized fisheye views*. ACM, 1986, vol. 17, no. 4.

[33] S. K. Card, G. G. Robertson, and J. D. Mackinlay, "The information visualizer, an information workspace," in *Proceedings of the SIGCHI Conference on Human factors in computing systems*. ACM, 1991, pp. 181–186.

[34] R. Spence and M. Apperley, "Data base navigation: an office environment for the professional," *Behaviour & Information Technology*, vol. 1, no. 1, pp. 43–54, 1982.

[35] S. Schreibman, R. Siemens, and J. Unsworth, *A companion to digital humanities*. John Wiley & Sons, 2008.

[36] R. M. Martins, V. Simaki, K. Kucher, C. Paradis, and A. Kerren, "StanceXplore: Visualization for the interactive exploration of stance in social media," in *2nd Workshop on Visualization for the Digital Humanities (VIS4DH'17), October 2017, Phoenix, Arizona, USA*, 2017.

[37] M. Kleppmann, *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media Inc., 2017.

[38] B. Laugwitz, T. Held, and M. Schrepp, *Construction and evaluation of a user experience questionnaire*. Springer, 2008.

[39] ISO, "ISO 9241: Ergonomics of Human System Interaction," International Organization for Standardization, Geneva, Switzerland, Tech. Rep., 2018.

[40] T. Jokela, N. Iivari, and V. Tornberg, "Using the ISO 9241-11 definition of usability in requirements determination: Case study," in *HCI*, 2004, pp. 155–156.

[41] M. Laitinen, J. Lundberg, M. Levin, and R. M. Martins, "The nordic tweet stream: A dynamic real-time monitor corpus of big and rich language data," in *Digital Humanities in the Nordic Countries (DHN 2018), 3rd Conference*, 2018.

[42] "The JavaScript Object Notation (JSON) data interchange format," RFC 7159, 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7159.txt

[43] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

A. Configuration Twitter Language Usage

```
{
  url: 'localhost:11219',
  name: 'Twitter Explorer',
  host: 'twitter.multiscale-explorer.de',
  fields: [
    { name: 'date', type: 'time' },
    { name: 'language', type: 'categorical' },
    { name: 'hasc2', type: 'categorical' }
  ],
  filter: {
    hasc2: { type: 'startsWith', domain: [] },
    language: { type: 'includes', domain: ['en', 'sv', 'no', 'fi'] },
    date: { type: 'between', domain: [+new Date(2017, 0, 1), +new Date(2017, 11, 31)] }
  },
  frontend: [
    {
      title: 'Language',
      widget: 'categories',
      width: 3,
      field: 'language'
    },
    {
      title: 'Origin',
      widget: 'map',
      topos: [
        'https://glcdn.githack.com/KRKnetwork/maps-scandinavia/raw/master/sweden-municipalities.json',
        'https://raw.githubusercontent.com/deldersveld/topojson/master/countries/norway/norway-municipalities.json',
        'https://glcdn.githack.com/KRKnetwork/maps-scandinavia/raw/master/denmark-municipalities.json',
        'https://glcdn.githack.com/KRKnetwork/maps-scandinavia/raw/master/finland-regions.json'
      ],
      levels: [],
      width: 6,
      field: 'hasc2'
    },
    {
      title: 'Tweets',
      widget: 'list',
      width: 3,
      field: 'text'
    },
    {
      title: 'Timeline',
      widget: 'timeline',
      width: 12,
      field: 'language',
      over: 'date',
      extent: [+new Date(2016, 10, 06), +new Date(2018, 1, 1)]
```

```
        }
    ]
}
```

B. Configuration Södra Forestry Yields

```
{
  url: 'localhost:11219',
  name: 'Soedra Explorer',
  host: 'localhost:11218',
  fields: [
    { name: 'time', type: 'time' },
    { name: 'speciesGroupName', type: 'categorical' },
    { name: 'numberOfLogs', type: 'categorical' },
    { name: 'hasc', type: 'categorical' }
  ],
  filter: {
    hasc: { type: 'startsWith', domain: [] },
    speciesGroupName: { type: 'includes', domain: ['Spruce', 'Pine', '
        Birch'] },
    numberOfLogs: { type: 'excludes', domain: [] },
    time: { type: 'between', domain: [+new Date(2017, 1, 1), +new Date
        (2017, 2, 1)] }
  },
  frontend: [
    {
      title: 'Species',
      widget: 'categories',
      width: 3,
      field: 'speciesGroupName'
    },
    {
      title: 'Logs per Tree',
      widget: 'categories',
      width: 3,
      field: 'numberOfLogs'
    },
    {
      title: 'Area',
      widget: 'map',
      topos: [
        'https://glcdn.githack.com/KRKnetwork/maps-scandinavia/raw/
            master/sweden-municipalities.json'
      ],
      levels: [],
      width: 6,
      field: 'hasc'
    },
    {
      title: 'Timeline',
      widget: 'timeline',
      width: 12,
      field: 'speciesGroupName',
      over: 'time',
      extent: [+new Date(2016, 10, 1), +new Date(2017, 4, 1)]
    }
  ]
}
```

C. Full Requirement List

Requirement 1 (Accessibility)
The system has to enable multiple users/researchers to look into the same dataset at the same time, without having to set up the system multiple times.

Requirement 1.1
The presentation has to be accessible across different devices without unnecessary overhead.

Requirement 1.2
The presentation has to be decoupled from the data source.

Requirement 2 (Adaptability)
The system has to be easily adaptable to different datasets regarding structure, data source, and viewing angle.

Requirement 2.1
The presentation has to be easily configurable to provide different views into a dataset.

Requirement 2.2
The system has to be easily adaptable to handle new data sources and data structures.

Requirement 2.3
The user has to be able to quickly swap between different setups.

Requirement 3 (Filterability)
The user must be able to filter the dataset from a selected viewing angle in the presentation, according to the view's unique presentation features.

Requirement 3.1
A filter applied in one viewing angle must be broadcasted directly to all other viewing angles.

Requirement 3.2
The user has to be able to filter the dataset down to a single event.

Requirement 4 (Multiscale)
The presentation has to provide the possibility of visually exploring datasets in multiple scales in every viewing angle that would benefit from it.

Requirement 4.1
The presentation has to provide a map view that employes multiscale techniques.

Requirement 4.2
The presentation has to provide a stacked time series view that employes multiscale techniques.

Requirement 5 (Usability)
The system has to follow general guidelines for usability.

Requirement 5.1
The presentation shall not include any unnecessary information or possibilities of interaction.

Requirement 5.2
The presentation shall avoid context disruption on interactions as much as possible.

Requirement 5.3
The amount of transfered data from the data source to the presentation shall be kept as light as possible.

Requirement 5.4
The presentation shall employ different, clear to understand colorschemes.

Requirement 5.5
The added value of the system as a whole for the data exploration shall be clear to users.

Requirement 5.6
The added value of the employed multiscale techniques shall be clear to users.

Requirement 5.7
The basics of interacting with the data through the presentation shall be easy to learn.

Requirement 5.8
Every possible action by the user shall be reproduceable for him.

Requirement 5.9
The user shall be enabled to solve possible tasks with the system after a short introduction into the presentation.