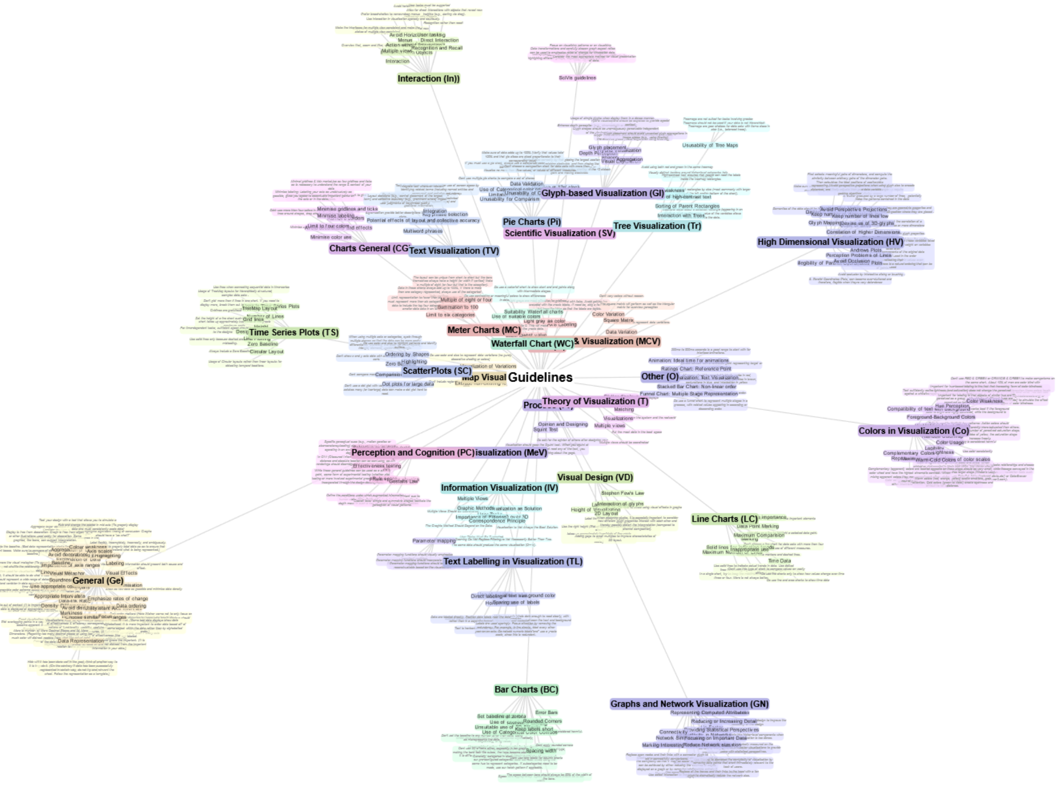
VisKnowsBest

A Web Catalogue of Visualization Guidelines and Best Practices

Bachelor's Thesis
12-07-2023

by Joe Müller, 19-735-299



Interaction (In)
Glyph-based Visualization (Gl)
Pie Charts (Pi)
Scientific Visualization (SV)
Tree Visualization (Tr)
High Dimensional Visualization (HV)
Charts General (CG)
Text Visualization (TV)
Time Series Plots (TS)
Meter Charts (MC)
Waterfall Chart (WC)
ScatterPlots (SC)
Guidelines
Other (O)
Theory of Visualization (T)
Colors in Visualization (Co)
Perception and Cognition (PC)
Visual Design (VD)
Information Visualization (IV)
Line Charts (LC)
General (Ge)
Text Labelling in Visualization (TL)
Bar Charts (BC)
Graphs and Network Visualization (GN)

Supervisors:
Prof. Dr. Renato Pajarola
Dr. Alexandra Diehl

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

University of Zurich^UZH

VISUALIZATIONANDMULTIMEDIALAB

Abstract

Visualization guidelines are one of the core elements of the visualization field. Guidelines pose an essential role for practitioners, researchers and students alike and therefore they should not just be accepted as they are proposed, but discussed, evolved and verified as thoroughly as possible. Currently, only a small amount of the proposed visualization guidelines go through this in-depth process, with the majority of them being the more prominent guidelines, such as the *Data-Ink-Ratio* or the *Chart-Junk debate*. While there are tools showcasing categorizations of visualization guidelines and forums for discussing them, there is no tool that acts as an accessible and easily navigable central source of knowledge for visualization guidelines. In the scope of this thesis, a tool is developed that provides guidance and context on visualization guidelines by providing additional information such as taxonomies and related scientific resources. To showcase the functionality of the tool, additional data required to provide guidance on visualization guidelines will be collected. The collected data is added to an existing collection of visualization guideline related data. The complete data will then be displayed on the VisKnowsBest tool to showcase a tool with the potential of acting as a central repository for visualization guidelines.

Acknowledgments

I would like to thank Dr. Alexandra Diehl for all the time she invested in this thesis. I really appreciate her availability even on short notice for any of my questions. Her guidance, advice and critical questions helped me in the course of writing this thesis. Her large knowledge of the visualization field and especially visualization guidelines and her vast experience were very helpful, as she was able to provide important inputs and literature recommendations to allow me to better understand concepts needed in the course of my thesis.

I would also like to thank the Visualization and Multimedia Lab (VMML) of the University of Zurich and its head, Prof. Dr. Pajarola, for allowing me to write this thesis in the field of visualization. Ünsal Satan was always there when any technical problems arose and provided me with helpful advice when it came to the implementation of the VisKnowsBest tool, therefore, my thanks also go to him. Sajan Srikugan also deserves my thanks, as his master's thesis was a huge inspiration for my bachelor's thesis and the taxonomy and data sourced and used in Sajan's thesis are being displayed in the VisKnowsBest tool. Finally, I would like to thank Prof. Min Chen from the University of Oxford for his valuable insights and recommendations at the start of my thesis.

Contents

List of Figures

Listings

List of Tables

1 Introduction

Visualization guidelines provide guidance to practitioners, novices and researchers alike. Precise and unambiguous guidelines provide help in creating clear and informative visualizations if applied in the right context. Thus, a bad or misleading guideline or a guideline employed in the wrong context can lead to plain bad or misleading visualizations. In the *Theory of Visualization*, proposed by Chris Johnson [Joh04] which was formalized and summarized by Chen et al., guidelines are also proposed to be part of an underlying framework for visualization [CGJ$^+$17].

Visualization guidelines can be found in books, scientific publications, blogs, websites and other forms of media. Their sheer number as well as the fact that there is no central hub for visualization resources can lead to situations where valuable resources are not discovered or used due to the overwhelming number of search results [LAC$^+$23]. While there are many collections of visualization guidelines, they often lack additional information and context, such as provenance data, evidence or a source. This makes it hard to distinguish whether what is being presented as a guideline is a personal preference, a preferred practice for a company or institution or a guideline which is based on empiric evidence and scientifically proven.

Creating a centralized repository for visualization guidelines, which provides provenance data, different scientific viewpoints and the public and scientific discourse offers a solution to these problems. While different publications highlight the need for such a tool, Scott-Brown goes one step further by proposing a structure, defining criteria which must be met and offering approaches to provide accessibility and searchability within such a repository [SB18].

In the scope of this thesis, I propose a web tool which functions as a visualization guideline repository. I analyse a preexisting set of over 200 guidelines, which provides additional data such as the guidelines' sources, categorizations and topics related to the guidelines. I further supplement the existing data set with additional data to populate the tool in order to showcase its functionality. The analysed guidelines are subjected to a keyword-generation process in order to provide additional context and aid searchability within the tool. Based on the keywords extracted from each guideline, related publications are made available for the user. I use a preexisting taxonomy which relates the guidelines based on categorization. The relationship between guidelines are then assigned a value based on their semantic similarity.

The structure of the web tool as well as the underlying database allow for easy extension and scaling. A clean and uncluttered interface is presented, which provides easy and intuitive navigation and searchability.

1.1 Motivation

In today's information space, the sheer number and variety of resources which are directly or indirectly related to visualization guidelines offer researchers, practitioners and students data and information on an unprecedented scale. While the large amount of knowledge and information about the field presents opportunities to work with and provides avenues for further research, it can also be overwhelming and intimidating, especially for novice users and practitioners. Next to the amount of information, the largely unstructured nature of it and the wide variety of different media they are presented in make it difficult to navigate the information space and to build understanding and a coherent mental model in the field of visualization guidelines. Finally, missing context and provenance data as well as contradicting information can create uncertainty and mistrust in the accessible information for people relying on it. All of these problems also provide motivation for creating a tool that aims to offer up solutions to them. With the creation of a scalable and extensible tool which has the potential to function as a central hub for knowledge and information related to visualization guidelines, I can address the problems of trustworthiness and uncertainty. My personal motivation for this thesis consists of learning about visualization guidelines. I want to understand the current problems posed in this field to propose a sustainable solution. Additionally, this thesis allows me to explore the options and possibilities of data analytics to create additional context for visualization guidelines. Finally, the thesis allows me to apply and deepen my software engineering skills by implementing a full stack application with an extendable and scalable architecture while learning an array of new technologies and frameworks.

1.2 Goals of the Thesis

The goals of this bachelor thesis are the following:

- **Creation of a Dataset to Populate the VisKnowsBest Tool:** Based on an existing collection of guidelines additional data, such as keywords and related resources, will be collected to provide guidance and context on the guidelines when displayed in the VisKnowsBest Tool. Additionally, for an existing taxonomy, the relationship between categories, subcategories and guidelines is visualized by analysing their semantic similarity based on keywords assigned to the guidelines.
- **Implementation of the Web Tool VisKnowsBest:** The main goal of the thesis is the implementation of the VisKnowsBest tool which serves as a prototype for a central knowledge repository for visualization guidelines. The tool's architecture is scalable and extendable whilst its core functionality provides an overview of visualization guidelines with details on demand. It allows users to interactively retrieve related publications for visualization guidelines and to explore the guidelines' context and categorization based on an existing taxonomy.

1.3 Contributions

The main contribution of this thesis will be the VisKnowsBest web tool which can be viewed as a prototype for a centralized source of knowledge for visualization guidelines. Next to functioning as a central repository for visualization guidelines, VisKnowsBest also offers users context such as the sources of the guidelines and related keywords and key phrases. Using the related keywords, additional related resources for the guidelines are fetched to provide additional guidance and context. The tool's main view presents a list consisting of an overview for each guideline while offering up a second view which provides details on demand for each guideline. The guidelines also are represented in the context of a taxonomy created and used in a masters thesis by Sajan Srikugan [Sri21] using three different representations:

- **Force-Driven-Graph Overview Visualization:** A Graph including all categories, subcategories and guidelines of the taxonomy. The graph displays the whole taxonomy complete with all guidelines
- **Extended Tree-Graph Overview:** A tree graph where categories and subcategories can be extended to show their contents. Provides an interactive, explorable presentation of the taxonomy, heavily inspired by the *VisGuidesExplorer* created in Sajan Srikugans master's thesis [Sri21].
- **Complete Graph displaying Semantic Similarity of Guidelines and Categories:** A collapsible list of the taxonomy that on the selection of a category or subcategory displays a complete force-directed graph of all direct children. The links between each set of children represent their semantic similarity with respect to their keywords.

The thesis adds additional data to an existing collection of guidelines, which was created in the scope of Sajan Srikugans master's thesis [Sri21]. The additional data is then used to allow users to interactively fetch related publications to provide additional guidance on a guideline. The additional data consists of keywords and -phrases generated and extracted using an automated process developed in the scope of this thesis. The keywords further are used to run a semantic similarity calculation on all guidelines and categorizations for an existing taxonomy.

1.4 Outline

This thesis begins with Chapter 2 on related work, where relevant context and underlying concepts are highlighted. The third chapter carries over this knowledge and focuses on establishing what is relevant data for the VisKnowsBest tool and gathering, calculating and processing it. The fourth chapter focuses on the technical side of the implementation of the VisKnowsBest tool, where architecture, frameworks and other details of the implementation are presented. The fifth chapter showcases the VisKnowsBest tool and highlights its functionality. The sixth and final chapter reflects on the process in this thesis and discusses shortcomings and future approaches to extend on this work and offers possible solutions and augmentations for limitations within this thesis.

- **Chapter 2: Related Work:** This chapter starts off by introducing the concept of the *Theory of Visualization* and highlighting its relevance and importance. It then discusses visualization guidelines, including problems with accessibility, trustworthiness and correctness of visualization guidelines, as well as missing structures for guidelines. Existing and proposed solution approaches for these issues then are presented. Guidance and especially designing guidance are summarized and techniques, algorithms and frameworks used in the data collection and calculation are presented.

- **Chapter 3: Data Collection and Preparation:** This chapter first establishes what data needs to be represented in the VisKnowsBest tool. As the data is based on an existing collection of guidelines, the missing data needs to be collected, calculated and processed. This includes the generation and extraction of keywords and key phrases for each guideline as well as gathering publications related to each guideline. In a second step, the collection and calculation of the semantic similarities between guidelines are explained. Finally, the preparation and formatting of the taxonomy-based data for the implementation of the graphs are presented.

- **Chapter 4: Implementation of the VisKnowsBest Tool:** This chapter focuses on the technical side of the implementation of the VisKnowsBest tool. It first presents the functional and non-functional requirements. It then focuses on the technical aspects of the implementation including the used frameworks and technologies as well as the software architecture and design decisions.

- **Chapter 5: VisKnowsBest: Presentation and Evaluation:** This chapter focuses on the VisKnowsBest tool and presents its functionality. It showcases the navigation and interaction with the tool. Finally, it evaluates the tool by the functional and non-functional requirements.

- **Chapter 6: Conclusion and Discussion:** The last chapter will summarize the work of the thesis and reflects on it. Based on the reflection, it addresses shortcomings, possible extensions to the VisKnowsBest tool and possibilities for future work.

2 Related Work

Guidelines are frequented when creating visualizations, as one of the main means of guidance and advice. Oxford English Dictionary defines guidelines as ”A rule, principle, or general statement which may be regarded as a guide to procedure, policy, interpretation, etc., or (especially) as giving authoritative guidance.” [Oxf23]. Guidelines can be valuable to researchers, practitioners and novices in the visualization field alike. For novices and practitioners, they can be seen as advice-giving and guidance tools when confronted with the task of creating visualization or in any other context of working with visualizations. For researchers guidelines represent the current point of knowledge and research, condensed down into a single guideline [DAREA$^+$18].

2.1 Theory of visualization

The *Theory of Visualization* was mentioned by Chris Johnson as being one of the top scientific research problems for visualization [Joh04]. More than a decade later, Chen et al. summarized the findings of theory-based panels and workshops at IEEE Visualization Conferences as well as other visualization-based events with arguments that further show the necessity of establishing the *Theory of Visualization*.

2.1.1 Theoretical advances in visualization

As visualization positions itself in between human- and machine-centric processes and the fact that many other scientific disciplines make use of visualization, theoretical advances in visualization leads to a multidisciplinary impact and advances in those fields. While theoretical research is sparse, compared to other fields, visualization can adapt and profit from progress made in other fields, where theoretical research is at a more sophisticated state. Additionally, theoretical research currently is the domain of only a few researchers, which is a further limitation [CGJ$^+$17]. Finally, it is argued that ”As in all scientific and scholarly subjects, theoretical development in visualization is a necessary and integral part of the progression of the subject itself.” [CGJ$^+$17, p.2].

2.1.2 Taxonomies and ontologies for visualization guidelines

Chen et al. highlight the central role of developing clear taxonomies and ontologies in a theoretic framework for visualization. Taxonomies and ontologies are essential for effective communication, understanding and organizing in any field of research [CGJ$^+$17]. There have been different approaches to developing taxonomies or classifications for visualization guidelines, such as [KL16, DKAR$^+$20, Sri21] which all analysed sets of visualization guidelines using Grounded Theory. Tory and Möller’s work focuses on creating a high-level taxonomy classifying visualization algorithms [TM04] while Shneiderman created a taxonomy based on tasks and data types [Shn96].

2.1.3 Importance of an open discourse

Chen et al. reiterate that the creation of an underlying theoretical foundation of visualization is not the sole responsibility of the minority of theoretical researchers in visualization but that it is the ”collective responsibility of the visualization community.” [CGJ$^+$17, p.14]. The community might be able to create a broader interest in theoretical visualization research by adjusting their expectations of novelty from groundbreaking discoveries to also welcoming incremental progress. [CGJ$^+$17] Additionally, building confidence and interest in future researchers to tackle these fundamental problems could be aided by tools such as VISGuides which provide an easy and accessible way to discuss visualization guidelines [DAREA$^+$18].

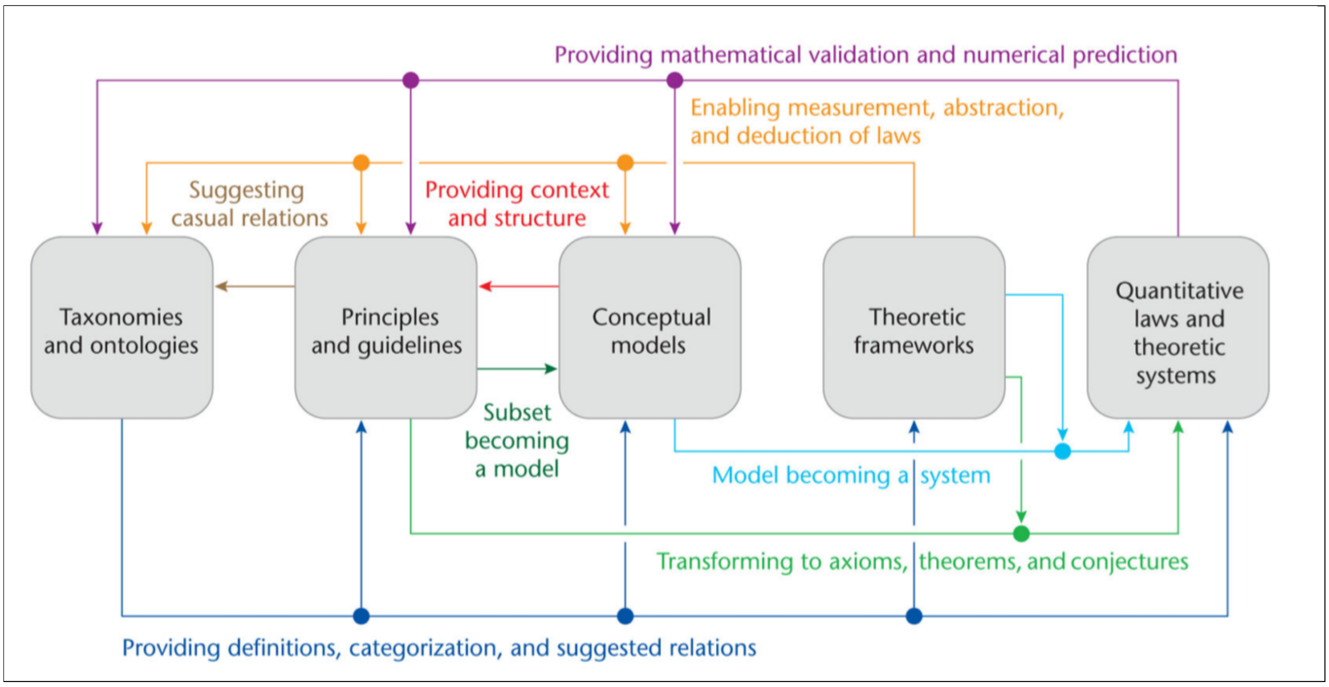

Providing mathematical validation and numerical prediction
Enabling measurement, abstraction, and deduction of laws
Suggesting casual relations
Providing context and structure
Taxonomies and ontologies
Principles and guidelines
Conceptual models
Theoretic frameworks
Quantitative laws and theoretic systems
Subset becoming a model
Model becoming a system
Transforming to axioms, theorems, and conjectures
Providing definitions, categorization, and suggested relations

Figure 2.1: Visualization of the iterative process for building a theoretical foundation for visualization. Displays the interaction of all aspects of the process and how they influence and benefit each other. Source: [CGJ+17]

2.2 Current problems for visualization guidelines

Currently, no complete collection of visualization guidelines exists. While attempting to create a complete collection of visualization guidelines borders on the impossible, an attempt can be made to create an extensive and searchable collection that functions as a central hub for visualization guidelines. Collections of guidelines are found in scientific publications [KW11, SSM11, BKC+13], sometimes collected as a by-product of working with visualization guidelines [KL16, DKAR+20, COSK21, Sri21], in books [Tuf01, Few12], and in different webpages, ranging from blogs [Inf, Flo, Kir], to websites providing advice on creating visualizations [IBM, Fin], discussion forums [Red, Sta, Quo, DAREA+] and collections of collections [Ces19].

2.2.1 Distribution of guidelines

Searching for something in the rapidly growing space of visualization resources does not always yield the best or even good results. The vast number and collections of such resources, stemming from researchers, practitioners, developers and students can be overwhelming and hard to navigate. Liu et al. offer up a collection of visualization resources [ACD+] as a helpful overview but they also highlight unsolved problems that are applicable to collections of visualization resources, such as timeliness and the literature explosion [LAC+23]. Scott-Brown highlights the broad and overall unstructured distribution of visualization guidelines, adding to the fact that visualization guidelines are accessible in different media, ranging from scientific publications, books, blogs, websites and many more. Visualization guidelines are also presented in different formats. Their presentation includes simple lists, decision trees, pattern libraries and continuous text [SB18]. Nguyen et al. mention the lack of systematically accessible guidelines as one of the key challenges for visualization in scientific publications and as one of the main reasons for faulty or misleading graphics [NJG21].

Choi et al. further see a problem in scattered guidelines as they often conflict with each other. The guidelines additionally often lack mentions of trade-offs in various use cases which can mislead people to use faulty guidelines in incorrect situations [COSK21]. Furthermore, Scott-Brown argues for a more structured and centralized collection of guidelines that functions as a reference tool, a teaching tool and a bibliography. To be able to fulfil all these functions a tool must be easily searchable and provide satisfying results [SB18].

2.2.2 Trustworthiness of visualization guidelines

Another problem visualization guidelines suffer from in their current environment, is, that they are often missing context, active discourse and provenance data. A guideline that is presented without context, additional information or provenance data should be met with scepticism and should be further investigated to ensure it isn't faulty or outdated [SB18, COSK21].

There exist many guidelines that are heavily debated among practitioners and researchers alike. A few examples of such guidelines and disagreements are the Chartjunk Debate [Few11] or the usage of the rainbow colour map [RT98, BTI07, FWD$^+$17]. Even though these disagreements and discussions are well known within the visualization research community, a practitioner or novice might find the originally proposed guideline and will decide to make use of it, due to not being aware that it might be outdated or even disproven.

2.2.3 Guideline structure

Diehl et al. mention a causal relation in the structure of guidelines in which an action (A) causes a consequence (B) while being exposed to a condition (C). For many guidelines the condition (C) still needs to be postulated, to situate the guideline in the right context [DARB$^+$22].

Kandogan and Le found three different forms of guidelines when analysing 510 guidelines using Grounded Theory. Most often (73%) guidelines are in the form of *declaratives*, which means they are statements regarding an opinion, design rule or principle. Secondly, they describe 16 % of the guidelines as *conditionals* that "state a condition and a consequence, which holds true only if the condition is satisfied". Finally, 11 % are in the form of *imperatives*, which often are similar to rules [KL16].

Choi et al. classified three categories of guidelines, according to eight attributes they defined. These categories consist of *problem-solving*, *enhancement*, and *alternative suggestion*. Additionally to the categorization, a structure template for guidelines is proposed, to help with finding, evaluating and creating guidelines [COSK21].

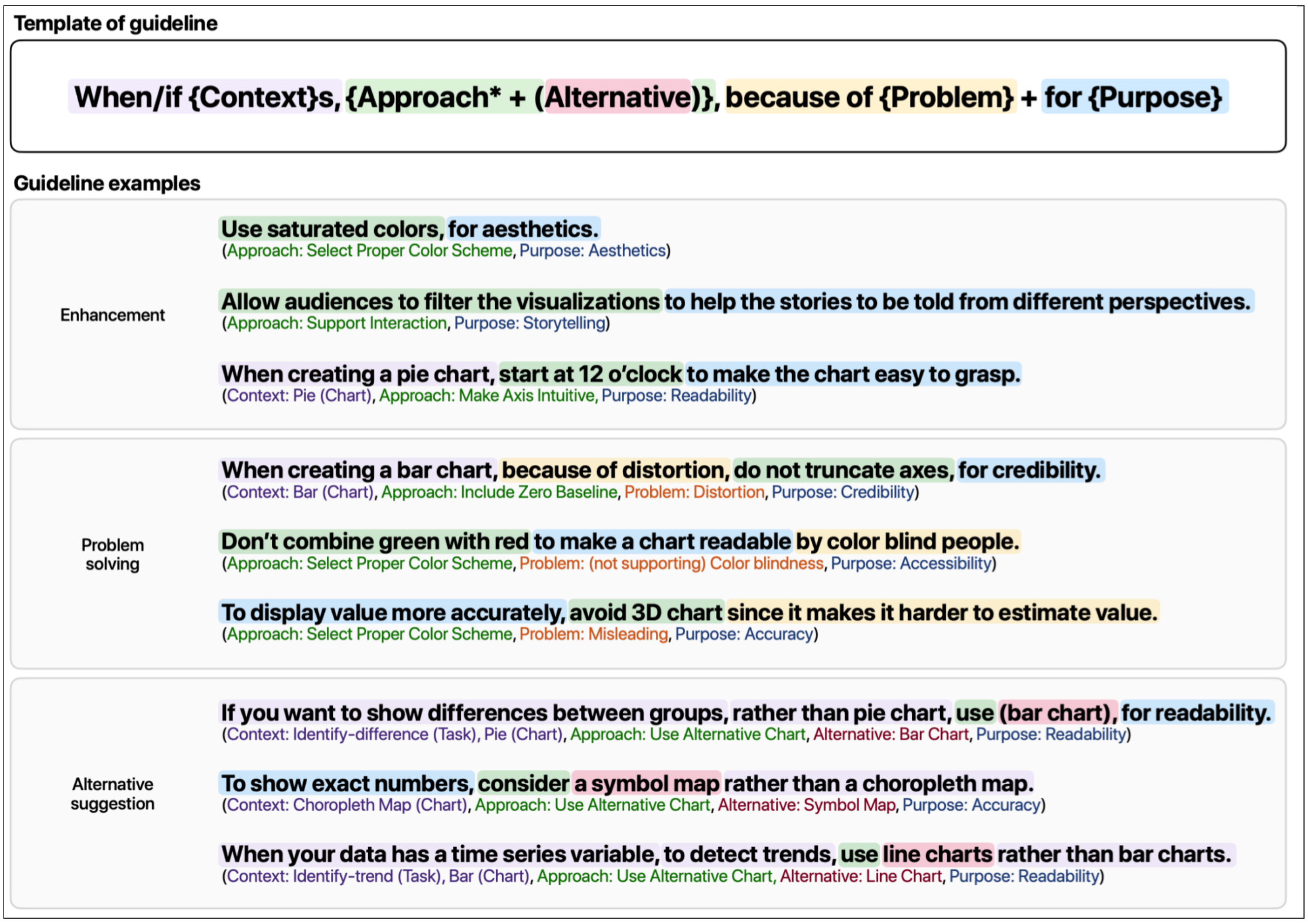
2.2.4 VISupply

The fact that there currently is no fixed or generally accepted process for creating, curating, updating and presenting visualization guidelines can lead to problems in creating guideline collections and maintaining them. Additionally, it is one of the sources for uncertainty about the applicability and validity of visualization guidelines. To approach this problem at its root Engelke et al. propose a process model based on a supply chain called *VISupply*.

The model provides an iterative approach which includes every aspect of a guideline's lifetime. Leaning on the supply-chain analogy, the model is split into *upstream* and *downstream* processes, where *upstream* processes include everything in the creation or production of a guideline, whereas the *downstream* processes mainly focus on the active use of guidelines, including the documentation of experiences with a guideline and retrieval and sharing of guidelines. The iterative nature of VISupply also allows for guidelines to constantly be updated and to take into account any new data or knowledge that might have an influence on a guideline. If the process of updating, verifying and, if necessary, even deprecating guidelines could be presented in some kind of provenance visualization it would offer a sense of security and certainty to any potential user of a guideline and their actuality. The authors mention, that VISupply can help bridge the gap between the people creating the guidelines and the people using them. VISupply as a conceptual model has the potential to constitute an important part of a larger system for data visualization. [EARC18]

2.2.5 Discussion forums for guidelines

As many people from different fields, ranging from practitioners to scientists and amateurs make use of visualization guidelines, many different contexts of use exist. Discussion platforms offer many benefits to the whole



Template of guideline

When/if {Context}s, {Approach* + (Alternative)}, because of {Problem} + for {Purpose}

Guideline examples

Enhancement	**Use saturated colors, for aesthetics.** (Approach: Select Proper Color Scheme, Purpose: Aesthetics) **Allow audiences to filter the visualizations to help the stories to be told from different perspectives.** (Approach: Support Interaction, Purpose: Storytelling) **When creating a pie chart, start at 12 o'clock to make the chart easy to grasp.** (Context: Pie (Chart), Approach: Make Axis Intuitive, Purpose: Readability)
Problem solving	**When creating a bar chart, because of distortion, do not truncate axes, for credibility.** (Context: Bar (Chart), Approach: Include Zero Baseline, Problem: Distortion, Purpose: Credibility) **Don't combine green with red to make a chart readable by color blind people.** (Approach: Select Proper Color Scheme, Problem: (not supporting) Color blindness, Purpose: Accessibility) **To display value more accurately, avoid 3D chart since it makes it harder to estimate value.** (Approach: Select Proper Color Scheme, Problem: Misleading, Purpose: Accuracy)
Alternative suggestion	**If you want to show differences between groups, rather than pie chart, use (bar chart), for readability.** (Context: Identify-difference (Task), Pie (Chart), Approach: Use Alternative Chart, Alternative: Bar Chart, Purpose: Readability) **To show exact numbers, consider a symbol map rather than a choropleth map.** (Context: Choropleth Map (Chart), Approach: Use Alternative Chart, Alternative: Symbol Map, Purpose: Accuracy) **When your data has a time series variable, to detect trends, use line charts rather than bar charts.** (Context: Identify-trend (Task), Bar (Chart), Approach: Use Alternative Chart, Alternative: Line Chart, Purpose: Readability)

Figure 2.2: Proposed Template for Guidelines with examples for each of the three proposed categories: Problem-Solving, Enhancement and Alternative Suggestion. Source: [COSK21]

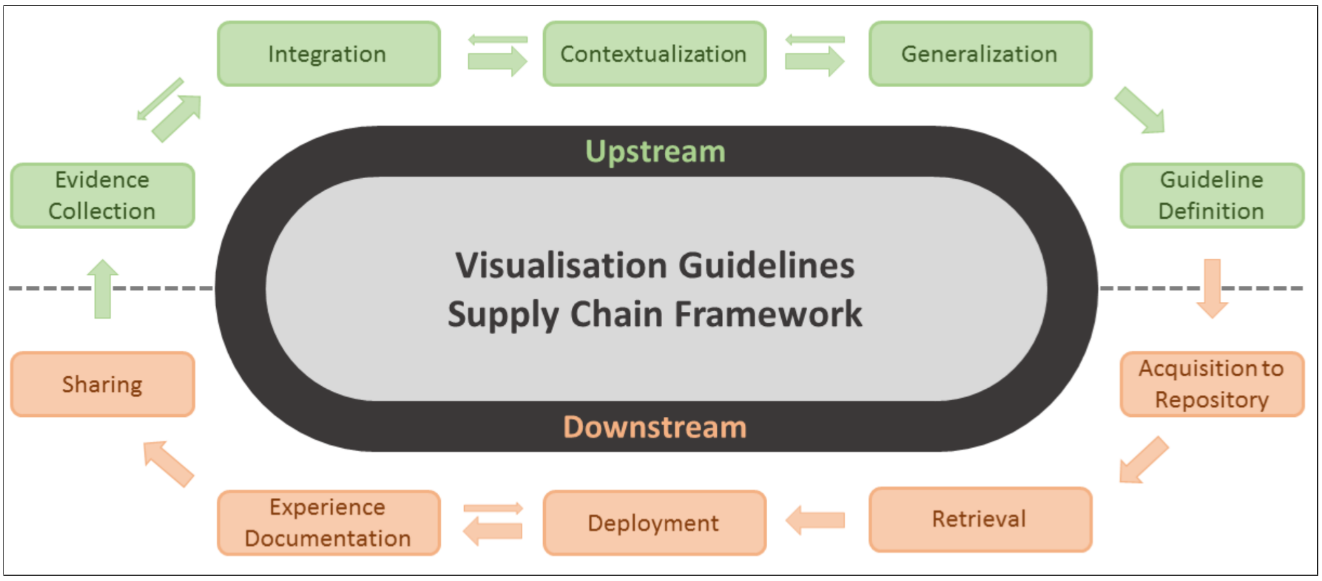


Figure 2.3: VISupply Model: A proposed supply-chain-based process model for creating, curating updating and presenting guidelines. Source: [EARC18]

community. It allows people to share their experiences with guidelines which other people can profit from. At the same time, these experiences and observations can be a starting point for the scientific analysis of guidelines which can lead to the evolution and generalization or restriction of guidelines and the creation of new guidelines. Finally, it offers people with uncertainty a platform to easily and widely find and provide help and therefore making guidelines more accessible [DAREA+18].

General discussion platforms

Discussions about visualization guidelines can be found on many general-purpose discussion forums such as Reddit [Red], Stackoverflow [Sta] or Quora [Quo]. While these forums allow open discussion about visualization guidelines, they are not specialized on either visualization or visualization guidelines. Reddit offers sub-forums like *r/DataIsBeautiful* [r/d] or *r/Visualization* [r/v] which have a more explicit focus on visualization.

Visguides

Visguides is an online discussion forum with its main and sole focus being visualization guidelines. Diehl et al. created this forum with the goals of collecting guidelines, encouraging the discussion of these guidelines and finally using the data sourced from the discussions to analyse guidelines and create formal knowledge about guidelines [DAREA+18]. In the first 18 months of being accessible, there have been 248 posts made to the forum, while it is important to mention that 148 of these posts have been made as a part of a student assignment [DKAR+20]. Currently, the most replies to a post on Visguides is eight and the highest number of views on a post is 6'000. There is a significant number of posts that have gotten no replies at all but several hundred views, which implies that there is interest in the topic of the posts but either the expertise or the motivation for creating a reply is lacking.

2.2.6 Coding visualization guidelines

There have been different approaches that contribute to the *Theory of Visualization*. Kandogan and Lee sourced about 550 guidelines from 18 different papers as well as five books. Using iterations of open coding as well as axial coding, they managed to derive 515 concepts, with 5 first-level concepts. Their work leaves out the last part of grounded theory, selective coding, which would be the final step in forming a larger theoretical scheme [KL16].

A different approach that also made use of grounded theory was conducted by Diehl et al. where they used 248 guidelines sourced from the online discussion forum *VisGuides*. The analysis and coding of these guidelines highlight the current lack of any generally accepted (or even proposed) categorization scheme for visualization guidelines [DKAR+20].

Srikugan gathered 204 guidelines from different sources (Peer-Reviewed, Blogs and more) and coded them manually. Keyword extraction methods (RAKE and Textrank) were further used in combination with open- and axial coding to group guidelines according to the keywords and -phrases. By applying an automatic topic modelling method, based on non-negative matrix factorization, final topic models as well as subcategories for these topics were generated which in turn are used to classify the guidelines in these topics and subtopics. The final result of this is visualized in the VisGuidesExplorer tool [Sri21].

All of these approaches to create a categorization of guidelines have the potential to aid the creation of an accessible and easily searchable browser or tool for visualisation guidelines. Any generally accepted topic categorization or coding scheme could potentially help with navigating and developing a guideline repository as they could contribute to the foundation of any tag-based search structure [SB18].

2.3 Guidance

Guidance is defined as ”help and advice about how to do something or about how to deal with problems connected with your work, education, or personal relationships” [Cam23] by the Cambridge Dictionary. When considering

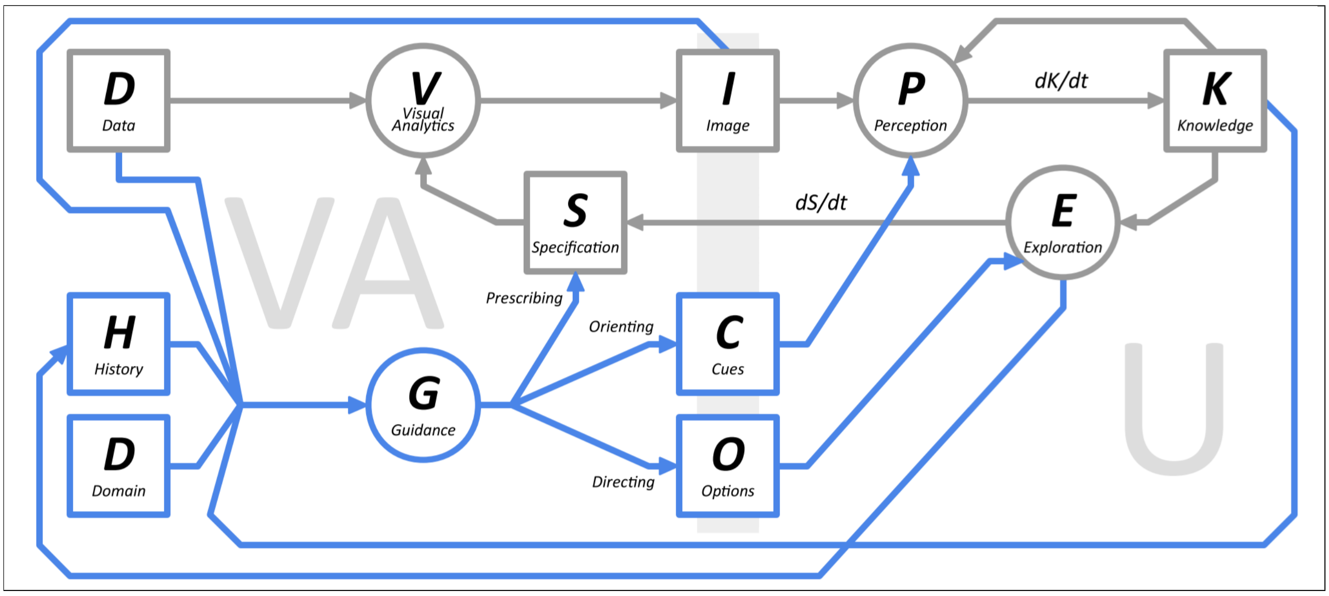


Figure 2.4: Model that combines van Wijks [vW06] model in grey with the components of guidance in blue. The left side represents aspects of guided visual analytics while the right side presents the user aspects of guidance. Source: [CGM+17]

.

guidance not generally but in the context of visual analytics, Ceneda et al. define it as ”Guidance is a computer-assisted process that aims to actively resolve a knowledge gap encountered by users during an interactive visual analytics session.” [CGM+17, p.2]. This definition presents guidance as a dynamic process that assists a user in any task or in a specific action of a task by offering up additional information or context which allows the user to then approach the sense-making on their own [CGM+17].

Ceneda et al. construct a model of guidance in visual analytics by building on van Wijks' model of visualization [vW06] and exchanging the term visualization with visual analytics. Adding on to this change they added guidance-related components to van Wijks' model to create a model of guidance [see Figure 2.4] in visual analytics [CGM+17].

2.3.1 Characterizing guidance

Schulz et al. also provide their characterization of guidance by defining four major aspects of guidance in visualization. These aspects can clarify the scope of guidance by defining the expected prior knowledge of a user (*Guidance Context*), what data, on which infrastructure and who interacts with it (*Guidance Domain*), the goal of what should be achieved (*Guidance Target*) and what amount of guidance is provided (*Guidance Degree)* [SSMT13].

Ceneda et al. create a similar characterization of guidance consisting of the three main categories *Knowledge Gap, Input and Output and Guidance Degree*. Knowledge Gap sub sums the type of guidance and the domain of the guidance [CGM+17]. These categorizations allow for a more systematic approach when designing guidance tools in visualization and in visual analytics. Pérez-Messina et al. propose a topology of guidance tasks, spanning the four dimensions of *why, how, what and when*. This model also raises the issue of upstream (system-provided) and downstream (user-provided) guidance and makes the observation that certain types of guidance may be favoured for some tasks or systems while others see no use at all, implying a two-dimensional guidance space defined by how observing and providing a system is. A system that provides little and observes little provides weak or even no guidance, while a system that observes the user while providing input itself offers co-adaptive guidance [PMCEA+22].

The process of co-adaptive guidance is introduced by Sperrle et al. where they focus on learning and teaching as the two central components. They define guidance with teaching intent as an actor that tries to change or

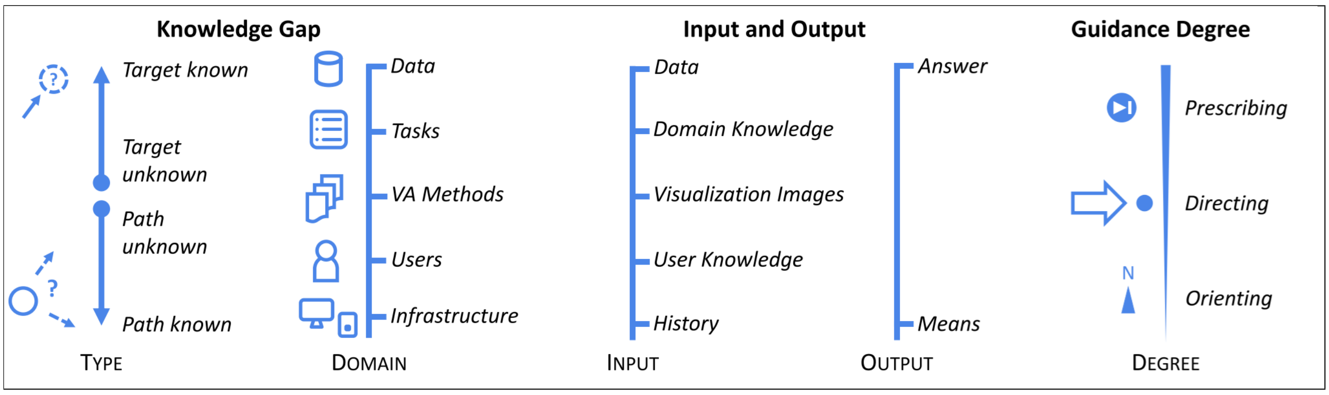


Figure 2.5: Characterization of guidance in terms of main aspects according to Ceneda et al. Main aspects of guidance include *Knowledge Gap*, *Input and Output* and *Guidance Degree*. Source: [CGM+17]

update another actor's mental model, by providing assistance, information or any other help, by taking an active role in the process. When an actor wants to verify their own model or hypothesis they make use of guidance with learning intent. The second actor is queried for additional information or a model to aid the first actor [SJB+20].

2.3.2 Designing guidance

Characterizing guidance in visualization and visual analytics helps with reasoning and communicating about the creation of guidance systems. While certain aspects of a guidance system could be designed by building on these frameworks, they all face the difficulty of choosing the right guidance actions as well as the right information to effectively close a user's knowledge gap [CGM+17, SJB+20]. To circumvent the difficulty of knowing and defining the users' expected knowledge gap, Sperrle et al. designed and implemented the library *Lotse*, inspired by Vega-Lite [SMWH17]. The guidance process that Lotse makes use of is based on guidance and inference loops proposed by Pérez-Messina et al. [PMCEA+22]. It works by defining different guidance strategies which are employed depending on the situation and the current state of the process and offers fitting suggestions [SCEA23].

Ceneda et al. propose a framework for designing guidance in visual analytics. The framework includes requirements for designing guidance as well as specific phases that should be included in the process. The requirements are based on earlier research by Ceneda et al. where they identified the necessary characteristics for effective guidance. These characteristics state that, above all, guidance should be effective, which can be achieved by it being available, trustworthy, adaptive, controllable and non-disruptive [CGM19].

Next to these requirements, Ceneda et al. identify four essential steps that should be completed while designing guidance. First, in the design process, you should clarify analysis goals as well as all phases that are expected to happen in the analysis. The second step focuses on identifying potential knowledge gaps of the users and what the appropriate information is to close that knowledge gap. In the third step, the characteristics of the guidance provided have to be considered and chosen according to the identified knowledge gap. In the fourth and last step the means to allow user feedback in order to fine-tune the provided guidance must be considered, to allow users to adapt the guidance to their specific needs [CAA+20].

2.4 Text analysis tools

To select, generate and curate the data to be displayed on the VisKnowsBest tool keyword extraction and semantic distance evaluation are needed.

2.4.1 Keyword extraction using KeyBERT

The BERT language representation model stands for *Bidirectional Encoder Representations from Transformers*. BERT is a pre-trained, bidirectional model, which allows for easy embedding by just adding one additional layer. This allows for the creation of state-of-the-art models for different natural language processing tasks that outperform other models while not relying on a heavy and complex task-specific architecture. The BERT model outperformed any previous results on eleven natural language processing tasks, including raising the GLUE score by 7.7% points to 80.5% [DCLT19].

KeyBERT [Gro21] is a minimal and easy-to-use Python library for keyword extraction. KeyBERT makes use of the precomputed BERT language model [Gro20]. For the keyword extraction itself, a passage of text of a phrase is fed to the BERT model to obtain the contextual feature vectors for each word in the provided text. The text passage then is assigned a vector by averaging out all word vectors and finally, the candidate keywords with the highest similarity to the average vector are returned according to

$$Sim_i = cos(w_i, W)$$

where Sim_i is the cosine similarity of w_i (word embedding vector of any word i) and W (text embedding vector). Key phrases then are created by selecting candidate keywords using the rule of adjacent keywords.

Using this approach the BERT-based keyword and key phrase extraction outperforms other extraction methods such as RAKE, SG Rank and TextRank on both, short sentences and long-paragraph text inputs [SL19].

2.4.2 Semantic similarity calculation using spaCy

SpaCy is a natural language processing library for Python that supports over 70 languages and offers pre-trained word vectors and provides methods for semantic similarity calculation. The similarity in spaCy is calculated by comparing word vectors using their cosine similarity

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}}$$

with $\mathbf{u}$ and $\mathbf{v}$ are the vectors and u_i and v_j are the i-th component of their corresponding vectors. When comparing strings containing multiple words or substrings, spaCy creates a sentence embedding vector by calculating the average of all word vectors contained within the string,

$$\bar{\mathbf{V}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{v}_i$$

where V_i are all the words contained within the sentence $\mathbf{V}$ and $\bar{\mathbf{V}}$ is the averaged out sentence embedding vector. SpaCy then again uses cosine similarity on the sentence embedding vectors to calculate their similarity [spa]. The word vectors used by spaCy are generated using the word2vec [MCCD13] algorithm.

3 Data Gathering and Preparation

This chapter focuses on obtaining, calculating, processing and preparing any data needed for the VisKnowsBest tool. First, it is established what data is necessary for the tool. In the next step, the keyword and -phrase generation and extraction process are showcased. Then the process for retrieving related resources for the guidelines are introduced. Finally, the front end methods and algorithms that prepare the taxonomy data for the implementation by running any necessary calculations and mapping the data into the required format for the graphs are presented.

3.1 Data collection and preparation

To showcase the main part of the VisKnowsBest tool which is implemented in this thesis, a sample set of guidelines and their related data must be collected. The main feature of the VisKnowsBest tool is providing guidance related to guidelines by providing additional information and context in the form of publications related to the guidelines. This set of additional data should be gathered in a way which minimizes bias. The data collection and processing are based on a set of over 200 guidelines that were collected in the course of Srikugans work [Sri21]. The existing set of data consists of the guidelines, sources which are categorized into primary-, secondary-, and tertiary sources, categorizations for the guidelines within a taxonomy and a topic to which each guideline is assigned.

3.1.1 Necessary data

The user should be provided with data related to the guideline to offer additional guidance while keeping the bias to a minimum. The additional data should also not be overwhelming to a user. Scott-Brown proposes that the necessary data for each guideline consists of the guideline itself, links to related guidelines, tags, external documents related to the guideline and access to discussions related to the guideline, specifically on the VisGuides forum [SB18].

The guidelines presented in the VisKnowsBest tool provide additional information based on Scott-Browns' propositions. The final available set of data per guideline in the VisKnowsBest tool includes the guideline, the guidelines source, topics assigned to the guideline, external related resources in the form of scientific publications and tags in the form of key phrases and keywords which are related to the guideline. Additionally, an existing taxonomy and categorizations are used to allow for a category-based exploration of the guidelines.

Finally, the guidelines provide links to related guidelines based on the taxonomy and are visualized in three different topical views within the VisKnowsBest tool. In contrast to Scott-Browns' proposition, the guidelines will not contain links to related discussions on the VisGuides forum, as there is no access point or API provided, which would be required for this.

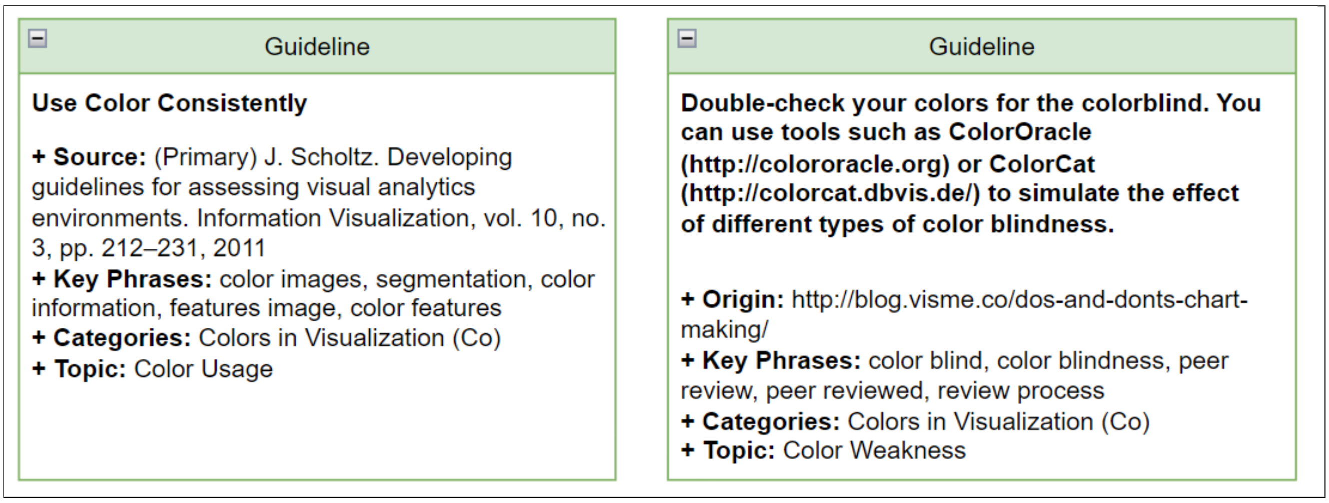
3.1.2 Key phrase extraction

The following libraries and APIs were used in the key phrase extraction process.

- **KeyBERT** is an easy-to-use Python keyword extraction method which is based on the BERT language model.
- **Semantic Scholar Academic Graph API(1.0)** provides access to over 200 million papers and 79 million authors. The API also provides additional metadata and AI-generated summaries for its resources. The API provides highly customizable queries and allows for conducting keyword searches.

Description	Category1	Category2	Guideline Topic (Manual)	Source
Don't use 3D effects either, especially in bar graphs. By making the bars look like cubes, the tops become obscured and it is difficult to discern where the top of the data really ends.	Bar Charts (BC)	3D-Visualization	Unsuitable use of 3D-Effects	(Primary) http://blog.visme.co/dos-and-donts-chart-making/
Limit representation to fewer than six categories. If you must represent more than six categories, consider truncating data to include the top four categories and then include smaller data sets in an 'other' category.	Meter Charts (MC)	-	Limit to six categories	(Tertiary) https://urbaninstitute.github.io/graphics-styleguide/
Rainbow colour Map is considered harmful	Colors in Visualization (Co)	Perception (PC)	Rainbow Colourmap	(Primary) D Borland and M T Russell II, Rainbow Color Map (Still) Considered Harmful, CG&A, 27(2), 2007.
Stephen Few's Rule #9 Avoid using visual effects in graphs	Visual Design (VD)	-	Stephen Few's Law	(Primary) Data Visualization 101: How to design Charts and Graphs
To improve value comparison use linear layout.	General (Ge)	Line Charts	Linear Layout	(Primary) Fuchs J.H. (2015). Glyph Design for Temporal and Multi-Dimensional Data: Design Considerations and Evaluation, (November).
Do not use multiple Pie Charts for comparison(Slices sizes are very difficult to compare side-by-side)	Pie Charts (Pi)	Animation	Unusability for Comparison	(Primary) The Wall Street Journal Guide to Information Graphics by Dona Wong (Dow Jones & Company 2010)

Table 3.1: Examples of collected guidelines and related data from Srikugan's work, on which the data collection in this work is based. Source: [Sri21]



Guideline

Use Color Consistently

+ Source: (Primary) J. Scholtz. Developing guidelines for assessing visual analytics environments. Information Visualization, vol. 10, no. 3, pp. 212–231, 2011
+ Key Phrases: color images, segmentation, color information, features image, color features
+ Categories: Colors in Visualization (Co)
+ Topic: Color Usage

Guideline

Double-check your colors for the colorblind. You can use tools such as ColorOracle (http://colororacle.org) or ColorCat (http://colorcat.dbvis.de/) to simulate the effect of different types of color blindness.

+ Origin: http://blog.visme.co/dos-and-donts-chart-making/
+ Key Phrases: color blind, color blindness, peer review, peer reviewed, review process
+ Categories: Colors in Visualization (Co)
+ Topic: Color Weakness

Figure 3.1: Examples of the data provided per guideline. The external resources are retrieved based on the key phrases and user preferences and are not saved to the database

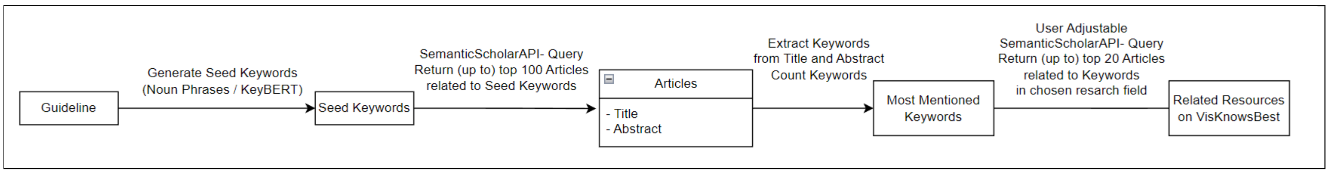

Guideline
Generate Seed Keywords (Noun Phrases / KeyBERT)
Seed Keywords
SemanticScholarAPI- Query Return (up to) top 100 Articles related to Seed Keywords
Articles
- Title
- Abstract
Extract Keywords from Title and Abstract Count Keywords
Most Mentioned Keywords
User Adjustable SemanticScholarAPI- Query Return (up to) top 20 Articles related to Keywords in chosen resarch field
Related Resources on VisKnowsBest

Figure 3.2: Automated keyword generation process and fetching of related resources

To automatically and interactively be able to load related resources in the VisKnowsBest tool, key phrases and keywords related to the guidelines must be extracted. These are then used to generate queries which access related resources utilising the Semantic Scholar Academic Graph API. The final, automated key phrase extraction process runs in Python and makes use of the Semantic Scholar Academic Graph API to access scientific publications and uses KeyBERT for the keyword and -phrase extraction.

Seed keyword generation In the first step of the keyword generation process, the seed keywords are generated. The seed keywords are extracted from the full text of the guideline using the KeyBERT method. KeyBERT allows for changing the length of keywords, removing stop words and only returning the top n most relevant keywords for an input. In the case of the seed keyword extraction the length of a key phrase is limited to one to three words and the amount of returned keywords and -phrases is based on the length of a guideline and is between two and five keywords. The seed keywords are limited to a maximum of five as more query parameters lead to too specific queries for the Semantic Scholar Academic Graph API which often returns empty or with too few results due to the specificity of the search. The seed keywords then are cleaned by removing any duplicates and multiples. The cleaned set of seed keywords then is used to create a query for the Semantic Scholar Academic Graph API to retrieve the necessary data for the next step in the keyword extraction process.

Listing 3.1: Seed keyword generation

```python
def seed_keywords_keybert(guideline):
    term_list = []
    searchterms = kw_model.extract_keywords(guideline, keyphrase_ngram_range=(1, 3),
        stop_words='english', top_n=max(2, min(5, len(guideline.split(" "))//2)))
    for term in searchterms:
        for item in term[0].split(" "):
            if item not in term_list:
                term_list.append(item)
    return term_list
```

Key phrase generation Using the generated seed keywords an API query to the Semantic Scholar Academic Graph API is made. The query returns the queried data for up to the top 100 most relevant publications for the provided seed keywords. In this first iteration, the queried data includes the title and abstract of the publications. All publications that either don't provide an abstract, title or neither are removed.

In the next step, the title and abstract are concatenated into a string. A keyword extraction using KeyBERT is run on the concatenated string for each publication. This extraction returns the top 15 keywords and -phrases for each publication with a length of one to three words per key phrase. The amount of times a keyword is mentioned for all title-abstract strings is counted and finally, the most mentioned keywords are assigned to the guideline. If no keywords are mentioned more than three times the guideline is assigned its seed keywords and is equipped with an additional flag that will be used to make users aware that the keywords assigned to the guideline have been assigned with a simplified process. The keywords and -phrases, together with the guideline title, source, categories and topics then are written into a CSV file which is then imported into the PostgreSQL database.

Listing 3.2: Keyword generation

```
for guideline in guideline_data_set:
    try:
        response = clean_data(sc_query(seed_keywords_keybert(guideline), 100))
        keywords = sort_keywords(extract_keywords(response))
        guideline_dict[guideline]["keywords"] = list(keywords.keys())
        guideline_dict[guideline]["generated"] = True
    except:
        seed_kw = seed_keywords_keybert(guideline)
        guideline_dict[guideline]["keywords"] = seed_kw
        guideline_dict[guideline]["generated"] = False

    with open('guideline_data_large.csv', mode='a', encoding='utf-8', newline='') as
        file:
        writer = csv.writer(file)
        writer.writerow(create_guideline_entry(guideline, guideline_dict,
            seed_keyword_guidelines))
```

3.1.3 Semantic similarity calculation

The semantic similarity of two guidelines is calculated using the Python library *spaCy*. *spaCy* offers large pre-trained language models that come with BERT-based embedded word vectors. Because of the way similarity scores are calculated using word vectors, the order of words in a string given as an input to *spaCy* does not affect the similarity score. This is relevant to the similarity score calculations as otherwise, the order in which the keywords are concatenated to a string would affect the similarity rating of guidelines. By using *spaCy* similarity calculation this problem is emitted and the order of the keywords is irrelevant. Additionally, this ensures that the similarity of the two guidelines based on their keywords is symmetric, which should be expected. So by looping all guidelines over a helper method that takes as input the sets of keywords concatenated to a string for two guidelines, the similarity score for each set of guidelines is calculated. The result of these calculations is saved in a JSON file and provided directly to the front end.

Listing 3.3: Method for calculating semantic similarity of two guidelines.

```
def semantic_distance(keywords_g1, keywords_g2):
    nlp = spacy.load('en_core_web_lg')
    distance = nlp(format_keywords(keywords_g1)).similarity(nlp(format_keywords(
        keywords_g2)))
    return distance
```

3.1.4 Gathering of related resources

The gathering of related resources for each guideline is achieved by fetching them interactively on the VisKnows-Best tool instead of saving them to the database. By not saving the related resources to the database this also ensures access to the most relevant and timely publications for each guideline and minimizes the risk of providing resources that become outdated. Additionally, it offers more interactive guidance to the user as it allows them to adjust the resources to their field of interest.

Definition of query parameters in front end The fetching of the related resources is based on the Semantic Scholar Academic Graph API. In the front end of the application, the user can change the query parameters for the request call to the API. The modifiable parameters are the maximum number of resources returned and the field of study the resources are in. The default values are set to include all fields of research and to return up to ten publications. Whenever a user loads a detail page for the guidelines, a get request is initiated by the front end, which then initializes a call to the API to fetch the related resources, using the user specified values as query parameters. If the user adjusts their query preferences another get request is initiated to update the resources.

Listing 3.4: OpenAPI specification for the back end call to fetch related resources.

```yaml
paths:
  /guidelines/{guidelineId}/sources:
    get:
      tags:
        - sources
      summary: gets sources for guideline
      description: get sources for guideline according to query
      operationId: getSources
      parameters:
        - name: guidelineId
          in: path
          description: API query for Semantic Scholar
          required: true
          schema:
            type: integer
            format: int64
        - name: limit
          in: query
          description: Maximum number of results to return (between 1 and 20)
          required: true
          schema:
            type: integer
            minimum: 1
            maximum: 20
        - name: field_of_study
          in: query
          description: Field of study for filtering sources
          required: false
          schema:
            type: string
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                oneOf:
                  - type: array
                    items:
                      $ref: '#/components/schemas/SourceDto'
        '400':
          description: api call failed
```

```

components:
  schemas:
    SourceDto:
      type: object
      properties:
        title:
          type: string
        url:
          type: string
        authors:
          type: array
          items:
            type: string
        year:
          type: integer
          format: int64
        abstract:
          type: string
        citationCount:
          type: integer
          format: int64
```

Fetching mechanism for related resources The actual call to the Semantic Scholar Academic Graph API is made in the back end of the application. The server listens for any get-resources requests from the front end. Once a get request is initiated the server reads the parameters from the request. Using the guideline ID provided in the request the keywords related to the guideline are fetched from the database. A helper method then generates the query string for the Semantic Scholar Academic Graph API. The call then is made by passing the generated query string to an Axios get request. The query string defines what data should be returned for each resource and it specifies the maximum amount of returned resources and the field of study these resources are in. The keywords that are assigned to each guideline are used as the query terms to ensure that the resources are related to the guideline while keeping any user-side bias at a minimum.

The resources returned from the API call then are passed back to the front end where they are displayed to the user in descending order based on the number of citations.

Listing 3.5: Get method in back end to fetch related sources

```
app.get('/guidelines/:id/sources', async function(req: any, res: any) {
    const id = req.params.id;
    const limit = req.query.limit;
    const selectedFields = req.query.field_of_study;
    try {
        const data = await repository.findById(id.toString())
        if (data !== undefined) {
            const keywords = data?.keywords ?? [];
            const response = await axios.get(guideline_service.queryCreator(keywords,
                limit, selectedFields)); // Make the API call using Axios
            const responseData = response.data.data;
            const sourceDtos = responseData.map((entry: any) => SourceDtoMapper.toDto(
                entry));
            res.send(sourceDtos);
        }
    } catch(error) {
        console.log(error);
        res.status(400).send({error: 'api call failed'})
    }
});
```

3.2 Cleaning and preparing of taxonomy data

The taxonomy used in this thesis is based on a JSON file that represents the taxonomy in a dictionary-like structure created in a master's thesis by Sajan Srikugan [Sri21]. The VisKnowsBest tool makes use of the taxonomy in different parts of the code while needing different subsets or format variations of the data. Due to the fact that each needed subset of data can be derived from the original taxonomy file and the complexity and additional problems that come with saving the taxonomy in the database as a relational structure, the file is provided directly to the front end in shape of a UTF-8 encoded JSON file. Using this approach any needed subset of the taxonomy can be computed based on the provided taxonomy file without having to worry about database access and loading times.

The only changes done to the JSON file to prepare the data for usage were cleaning out any encoding errors, removing duplicates in the data set and finally encoding it into UTF-8 encoding to ensure that no errors occur when reading the taxonomy into the react application.

3.2.1 Preparing taxonomy data for graphs

The taxonomy was used in a variety of different ways. The VisKnowsBest tool implements three taxonomy-based views, which all somehow represent the taxonomy based on a hierarchical tree. This is then visualized in the form of an expandable list, an explorable hierarchical tree graph or a force graph with a tree structure representing the whole taxonomy. Additionally, one taxonomy-based view presents a complete graph of the set of direct children of a category or subcategory within the taxonomy. For either one of these representations, the graph must be presented in a form that includes a list of all nodes and a list of all links between nodes, which are represented as a tuple of starting node and goal node. Additional fields like weight or colour can be assigned to the nodes and links if it is necessary for the visualization.

Data preparation for hierarchical tree structures To provide the taxonomy in the needed format of a list of nodes and a list of links a recursive tree traversal is used. The taxonomy data set is traversed, starting at its root, and adds each traversed node to the list of nodes. For each recursive call of the tree traversal, the parent and child nodes are added as a tuple, which represents a link, to the link list. The stopping condition for the recursive call is a check on whether any children exist in the original taxonomy data set. The list of nodes and list of links then are accessed by any other front end method that makes use of them for a graph.

Listing 3.6: Simplified pseudocode of data preparation for hierarchical tree graph

```

function formatDataHierarchicalTreeGraph(node):
  nodeList = []
  linkList = []

  function traverseTree(node, level):
    nodeList.add((node.name, node.id, level))
    if (node.hasChildren()):
      for each child in node.children:
        linkList.add((nodeId, childId))
        // recursive call of traversal
        traverse(child, level + 1)
  return (nodeList, linkList)
```

Data preparation for complete graph To prepare the taxonomy data for the complete graphs of direct children for a node, first, a tree traversal is used to find the parent node. When the parent node is found all of its direct children are added to a list of nodes and the connection between each set of children is added to the graph and is assigned a weight which represents the semantic distance between two nodes. The semantic similarity data is available in a precomputed JSON file containing the semantic similarity between each set of guidelines. If the selected nodes A and B are not leaf nodes the semantic similarity is calculated by recursively calculating the average semantic similarity of all of their children

$$similarity_{AB} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} distance(a_i, b_j)$$

where a_i is the i-th child of A and b_i is the i-th child of B and n is the number of children for A and m is the number of children for B.

Listing 3.7: Simplified pseudocode of data preparation for complete graph

```

function formatDataCompleteGraph(node):
  nodeList = []
  linkList = []

  function findNode(rootNode, searchedName):
    if rootNode.children:
      for child in rootNode.children:
        if (child.name == searchedName):
          return child;
        else:
          searchedNode = findNode(child, searchedName)
          if (searchedNode):
            return searchedNode

  function calculateDistance(node1, node2):
    if (node1.isLeaf && node2.isLeaf):
      //returns distance from precomupted data et
      return distanceData(node1, node2)
    else:
      for every child1 in node1.children and every child2 in node2.children:
        return calculateDistance(child1, child2)

  return (nodeList, linkList)

  parentNode = findNode(node)

  for child1 in parentNode.children:
    nodeList.add(child1.name, child.id)
    for child2 in parentNode.children:
      if child2.id > child1.id:
        linkList.add(child1, child2, calculateDistance(child1, child2)

  return(nodeList, linkList)
```

4 Implementation of the *VisKnowsBest* tool

The core part of this thesis focuses on creating a web catalogue, called VisKnowsBest, of visualization guidelines and best practices to provide guidance on these subjects. The VisKnowsBest tool functions as a web repository for visualization guidelines by making them easily searchable and aiding the users' decision-making process by providing guidance in the form of additional literature on the current scientific discourse as well as keywords, sources and categorizations related to the guideline. This chapter focuses on the technical side of the implementation of this tool. This includes defining functional and non-functional requirements, presenting the choice and use of frameworks and libraries and showcasing and explaining the choice of software architecture.

4.1 VisKnowsBest requirements

The following requirements must be met by the VisKnowsBest tool:

Functional requirements

- A scrollable list representing an overview of all guidelines should be visible on the home screen.
- Selecting a guideline in the guideline list navigates to the guidelines details page where additional information (author, origin, additional resources) for the guideline is displayed.
- A collapsible and expandable hierarchical tree graph representing the taxonomy allows users to explore the taxonomy interactively.
- A search bar allows users to conduct text searches for the guidelines.

Non-functional requirements

- All guidelines together with their provenance data should be accessible to the user on selection with loading times being under one second.
- The UI is clean and clutter free.
- The UI provides simple and intuitive navigation for a user.

4.1.1 Additional requirements

The following requirements extend the VisKnowsBest tool by adding functionality or increasing the quality of the tool. These requirements got added to extend the minimal viable product of the VisKnowsBest tool.

- A graph representing the semantic similarity of categories and guidelines within the chosen taxonomy should be implemented to provide a user with additional context about the taxonomy.
- Options for the user to adjust the search results to search by source, categories and keywords should be available to allow users to tailor the text search to their needs.
- Options for the user to adjust the additional resources to their needs by changing their field of research and how many resources are displayed.

4.2 Implementation

The tool implemented in the course of this thesis should be viewed as a prototype or a minimal viable product of a potential final implementation of a web catalogue that functions as a central source of knowledge for visualization guidelines. The web catalogue is developed with a focus on scalability and extendibility, making it as simple as possible for future projects to extend the catalogue in terms of additional functionality or scale in terms of data made accessible.

4.2.1 Technologies

In the following subsection, the most important technologies, frameworks and libraries used during the implementation of the VisKnowsBest tool are presented.

Front end

The front end stack is based on ReactJS and makes use of a React Redux store. Any back-end calls are done by making use of redux-sagas. The front end additionally utilizes a generated API-Service which is generated by OpenAPI, based on a API contract, to ensure compatibility to the back end.

ReactJS React is an open-source framework for JavaScript. React allows for the creation of reusable UI components. By focusing on the creation of reusable UI components ReactJS ensures readability as well as scalability. To ensure optimized performance ReactJS makes use of a virtual DOM (Document Object Model) which minimizes direct manipulation of the DOM.

React Redux React Redux is a JavaScript framework for state management in React or other JavaScript frameworks. React Redux provides an application with a global store, that allows components to fetch and update data across an application while minimizing the unnecessary passing of props. React Redux synergizes with the Redux side effect manager Redux-Saga, which allows for handling any side effects such as making API calls or any computations or data manipulations.

Back end

The back end of the VisKnowsBest tool is implemented using Node.js and connected to the front end through an OpenAPI generated API-service and connected to a PostgreSQL database for storage.

Nodes.js Node.js is an open-source JavaScript framework for the creation of server-side web applications. Node.js also provides a large number of modules and packages through npm (Node Package Manager), which allows for easy introduction and usage of existing libraries.

OpenApi OpenAPI, formerly known as Swagger, provides a standardized way to describe and document RESTful API's. The API is described as a contract in either a JSON or YAML file. The description includes HTTP methods with their input parameters, error codes and any potential DTOs (Data Transfer Objects). Out of this specification, the interactive API documentation and any related boilerplate code needed for an API are generated.

Database

The database used in the VisKnowsBest tool is a PostgreSQL database that is connected to the back end and initialized by the use of node-pg.

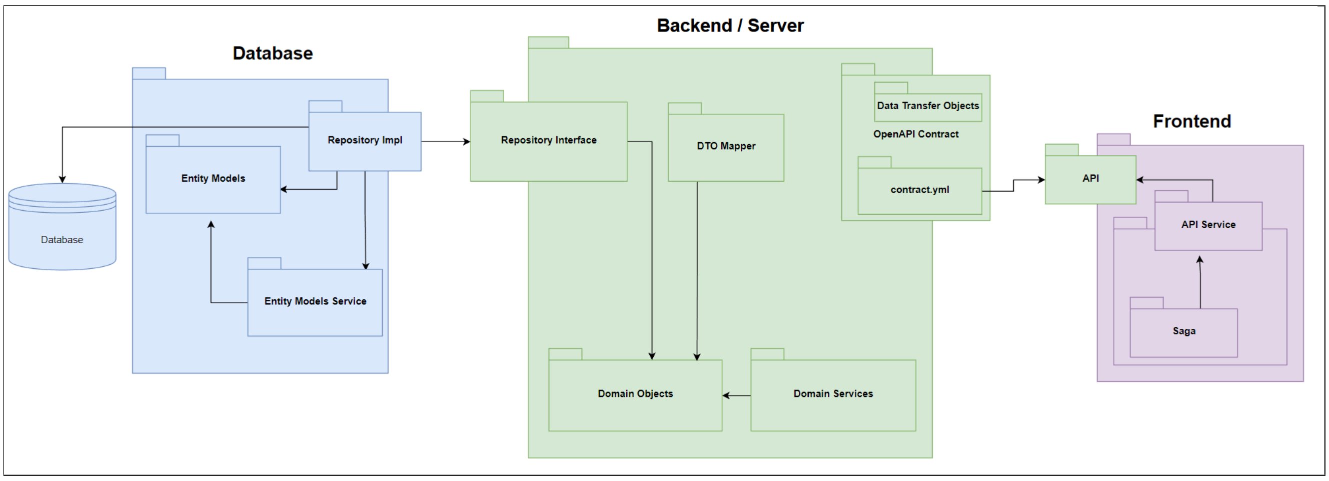


Figure 4.1: Package diagram of VisKnowsBest architecture (simplified)

PostgreSQL PostgreSQL is an open-source relational database system that offers support for JavaScript and Node.js.

Node-PG Node-Postgres or node-pg contains a collection of node.js modules for interacting with a PostgreSQL database. It supports a rich set of functionalities, including connection pooling, promises and async/await functionality.

4.2.2 Architecture

To ensure extendibility low coupling between different parts of the architecture must be enforced. For a project of this scale, the *Hexagonal Architecture* or *Port and Adapter Pattern* is suited ideally. This architecture pattern was proposed by Alistair Cockburn in 2005 [Coc05]. The Port and Adapter Pattern is based on an application core which contains the business logic of the application. The application core provides ports, which any external system can connect to, using a correctly implemented adapter. This ensures low coupling between components, as their only point of coupling are their respective ports and adapters.

The VisKnowsBest web catalogue consists of three major components: a database, a front end and a back end [see Figure 4.1]. The back end makes up the core of the application and provides ports for the front end and database. The front end port is provided in the shape of an API contract. Any service that consumes this API contract therefore automatically implements an adapter to this port. The database port is presented in the form of a repository interface.

Front end port - OpenAPI specification The front end port is defined as an API contract generated using OpenAPI specifications. OpenAPI specification allows for a standardized way of defining an API while also generating all related boilerplate code. Having the specifications for the whole API and any related DTOs (Data Transfer Objects) in one file also minimizes the risk of bugs and makes debugging easier as the only potential source for API-related bugs is in the specification file.

Listing 4.1: OpenAPI specification file contract.yml

```
    openapi: 3.0.3
info:
  title: VisKnowsBest- OpenAPI 3.0
  description: |-
    RestAPI for the VisKnowBest Tool.
servers:
  - url: http://localhost:5001/
paths:
  /guidelines:
    get:
      tags:
        - guidelines
      summary: get all guidelines
      description: get all guidelines
      operationId: getGuidelines
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/GuidelineDto'
        '400':
          description: No guidelines found
  ...
  /guidelines/{guidelineId}:
    get:
      tags:
        - guidelines
      summary: gets guideline by id
      description: get guidline by guideline id
      operationId: getGuidelineById
      parameters:
        - name: guidelineId
          in: path
          description: id of the guideline
          required: true
          schema:
            type: integer
            format: int64
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/GuidelineDto'
        '400':
          description: Invalid Id
        '404':
          description: Guideline not found
    ...
components:
  schemas:
    GuidelineDto:
      type: object
      properties:
        id:
          type: integer
          format: int64
        guideline:
          type: string
        keywords:
          type: array
          items:
            type: string
        generated:
          type: boolean
        origin:
          type: string
        categories:
          type: array
          items:
            type: string
        topic:
          type: string
      required:
        - guideline
        - generated
```

Database port - repository interface The database port is implemented by making use of a repository interface which defines all interactions between the back end and any database. Any adapter that implements the repository interface can connect a database to the back end. Currently, the repository interface consists of Create, Read, ReadAll, Update and Delete methods. In the current version, the tool only makes direct use of the Read and ReadAll methods, but a full set of CRUD operations is provided to ensure easy extendibility by defining a full set of CRUD operations for the database and front end.

Listing 4.2: Repository interface

```
interface GuidelineRepository {

    findAll(): Promise<Guideline[]>;

    findById(id: string): Promise<Guideline | undefined>;

    create(guideline: Guideline): Promise<Guideline | undefined>;

    update(guideline: Guideline): Promise<Guideline | undefined>;

    delete(id: string): Promise<boolean>;
}
```

4.2.3 Database implementation

The implementation of the database consists of two main parts: the initialization and setup of the database and the connection management between the database and the back end of the application.

Initialization of database The PostgreSQL database used in the application allows for an easy setup and makes use of the db-migrate package [KG] for initialization. DB-migrate enables the running of SQL scripts to set up the database, allowing for easy initialization of the database. By creating a function to run the initialization (seen in Listing 4.3) of the database and linking it to the file containing the SQL code that initializes the database, the initialization procedure can be modified by only modifying the code within the SQL file.

Listing 4.3: Initalization method for database

```
exports.up = function(db) {
  var filePath = path.join(__dirname, 'sql' , '20230330072156-initialize-up.sql');
  return new Promise( function( resolve, reject ) {
    fs.readFile(filePath, {encoding: 'utf-8'}, function(err,data){
      if (err) return reject(err);
      console.log('received data: ' + data);
      resolve(data);
    });
  })
  .then(function(data) {
    return db.runSql(data);
  });
};
```

Database connection The connection to the database is maintained by making use of the node-postgres (node-pg) package. When starting up the back end a PostgreSQL-Client is instantiated which then pools the database connections by specifying the connection details. This ensures that the client always has a pool of connections from the database to the back end. By pooling the connections there is no need to open, maintain and close a connection manually for each database request. Each time a request is sent to the database any free connection within the pool of connections is assigned to the request for the duration needed to fulfil the request.

Listing 4.4: Setup for database connection

```
const { Pool } = require('pg')

class PostgresClient {

    private pool;

    constructor() {
        /*
        Pool lazily manages connections and disconnection
        This way it isn't necessary anymore to manually connect
        Just pass the connection parameters in the instantiation
         */
        this.pool = new Pool(
            {
                host: "localhost",
                user: "postgres",
                port: 5432,
                password: /*password*,
                database: "postgres"
            }
        )
    }

    public async query(sql: string, params: any[] = []) {
        try {
            const result = await this.pool.query(sql, params);
            return result;
        } catch (error: any) {
            console.error('Error in Database Query: ${error}')
        }
    }
}

export default PostgresClient;
```

4.2.4 Implementation of taxonomy based views

The taxonomy-based views provide different graphical representations of the same taxonomy. To create the graphs representing the taxonomy the VisKnowsBest tool makes use of the *react-force-graph* library and the *react-d3-tree* library. Both these libraries provide simplified interfaces for implementing *D3.js* graphs in a react application.

- **react-force-graph:** The react-force-graph is a react binding for a web component to represent 2-dimensional force-directed graphs. It makes use of HTML5 canvas to render the graphs and uses a physics engine based on *d3-force* which uses a simplified velocity Verlet numerical integrator to simulate the physical interaction between particles or nodes [Ast].

- **react-d3-tree:** React-d3-tree is a react component for representing hierarchical data in an interactive tree graph based on D3's tree layout. The react-d3-tree provides a minimal setup interface and allows for easily making use of D3's tree layout in React [Kre].

Full hierarchical tree graph view The full hierarchical tree graph was implemented by making use of the *react-force-graph* library. The taxonomy data needed as input for the graph, which includes nodes and links, is provided to the graph in the form of a list containing all nodes and a list containing all links. The method providing the data in the correct format also assigns each node a colour depending on their parent such that every main branch is assigned a colour and each level within the hierarchy is assigned a more transparent version of the colour. This allows the user to easily distinguish different branches as well as the hierarchy within a branch.

Listing 4.5: Implementation of the hierarchical tree graph of the taxonomy as a force graph

```
<ForceGraph2D
  width={width()}
  height={graphFrameHeight()}
  graphData={createDataTree(taxonomyData, root, 3)}
  nodeAutoColorBy="group"
  nodeCanvasObject={(node, ctx, globalScale) => {
    const maxChars = 60;
    const label = decodeUTF8String(addLineBreaksGraph(node.name, maxChars));
    const fontSize = 3;
    const borderRadius = 6;
    const padding = 10;

    ctx.font = createFont(node);

    // Calculate the dimensions based on the modified label
    const lines = label.split('\n');
    const lineHeight = getFontSize(node) * 1.2;
    const maxLineWidth = Math.max(...lines.map(line => ctx.measureText(line).width));
    const bckgDimensions = [maxLineWidth, lines.length * lineHeight].map(n => n +
        getFontSize(node) * 0.2);

    ctx.fillStyle = node.color; //assignes the nodes with correct color
    ctx.beginPath();
    ctx.moveTo(node.x - bckgDimensions[0] / 2 + borderRadius, node.y - bckgDimensions[1]
         / 2);
    ctx.lineTo(node.x + bckgDimensions[0] / 2 - borderRadius, node.y - bckgDimensions[1]
         / 2);
    ctx.quadraticCurveTo(node.x + bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2,
         node.x + bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2 + borderRadius);
    ctx.lineTo(node.x + bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2 -
        borderRadius);
    ctx.quadraticCurveTo(node.x + bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2,
         node.x + bckgDimensions[0] / 2 - borderRadius, node.y + bckgDimensions[1] / 2);
    ctx.lineTo(node.x - bckgDimensions[0] / 2 + borderRadius, node.y + bckgDimensions[1]
         / 2);
    ctx.quadraticCurveTo(node.x - bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2,
         node.x - bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2 - borderRadius);
    ctx.lineTo(node.x - bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2 +
        borderRadius);
    ctx.quadraticCurveTo(node.x - bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2,
         node.x - bckgDimensions[0] / 2 + borderRadius, node.y - bckgDimensions[1] / 2);
    ctx.closePath();
    ctx.fill();

    ctx.textAlign = 'center';
    ctx.textBaseline = 'middle';
    ctx.fillStyle = getColor(node);


    const initialY = node.y - ((lines.length - 1) * lineHeight) / 2;

    lines.forEach((line, index) => {
      const y = initialY + index * lineHeight;
      ctx.fillText(line, node.x, y);
    });

    node.__bckgDimensions = bckgDimensions;
  }}

  nodePointerAreaPaint={(node, color, ctx) => {
    ctx.fillStyle = color;
    const bckgDimensions = node.__bckgDimensions;
    bckgDimensions && ctx.fillRect(node.x - bckgDimensions[0] / 2, node.y -
        bckgDimensions[1] / 2, ...bckgDimensions);
  }}
/>
```

The implementation makes use of the force component of the force graph via the default settings of the react-force-graph library. The default values assign each node the mass of *1* and each link the same constant for attraction between two connected nodes. The calculations for the repulsion of the nodes are based on a velocity Verlet integration. The links can be regarded as springs which are pulling nodes together. The force applied by the links is based on an implementation of Hooke's law. By making use of the default setting of the react-force-graph library the branches form clusters of connected nodes, which repulse other branches based on the size of their clusters. This leads to the effect of larger clusters being repulsed further from the centre of the graph while smaller clusters or branches stay closer to the centre of the visualization.

List and complete graph implementation The taxonomy list and complete graph view have the most complex implementation of all taxonomy-based views in the VisKnowsBest tool. The list presented on the left side of the screen makes use of the *material-ui* TreeView component. The list presents the taxonomy and is collapsible and expandable by clicking on any item in the list. By clicking on an item in the list the current selection and therefore the complete graph is reloaded to display the complete graph of children for the currently selected item.

The complete graph of the direct children of the current selection is again an implementation making use of the react-force-graph. The goal of this graph is to represent the semantic similarity of two children within a category. This is done by adjusting the width and length of the connecting links between two nodes, depending on their semantic similarity. This involves modifying the parameters of the simulation engine for the force feedback graph. The main change needed is to adjust the pull factor of the links to pull more closely related nodes closer together while allowing less closely related nodes to distance themselves further from each other. As the nodes repulse each other according to Coulomb's law their repulsion is inversely proportional to the square distance r between the two nodes, which is represented by their link. This results in the nodes exerting a stronger force on each other the closer they are. The links only exert a pull on the two nodes based linearly on their displacement from their relaxed position of the link, if no force was acting on the nodes, according to Hooke's law. This leads to the problem that if the length of the links is assigned linearly based on the similarity of two nodes, the repulsing force acting between the two nodes overpowers the pulling force acting on the link. This becomes more prevalent for nodes that are more similar and therefore closer to each other. The result of this would be that the distance between more similar nodes is less distinguishable than distances between nodes that are not similar to each other, as the repulsion falls off rapidly for nodes that are further apart. Therefore the repulsion force overpowers the pulling force for closer nodes while it becomes negligible for nodes that are further apart. To counteract this problem, the distance for the links between two nodes is assigned based on the inverse of the square of the similarity score, with the similarity score always being between one and zero. The distance is then additionally scaled by a linear factor to ensure that nodes do not overlap within the graph.

Listing 4.6: Implementation of the complete graph displaying semantic similarity of guidelines

```
<ForceGraph2D
  ref={fgRef}
  width={widthCompleteGraph()}
  height={completeGraphFrameHeight()}
  graphData={createCompleteGraphData(taxonomyData, root, threshold)}
  nodeAutoColorBy={node => mapNodeToColor(node, node.id)}
  linkWidth={link => link.distance * 5}
  onNodeClick={(node) => handleClick(node)}
  linkCurvature={0.1}
  minZoom={3}
  maxZoom={14}
  linkColor={() => {
    const opacity = 1;
    const color = '#333333';
    return `${color}${Math.floor(opacity * 255).toString(16)}`;
  }}
  nodeCanvasObject={(node, ctx) => {
    const maxChars = 25;
    const label = decodeUTF8String(addLineBreaksGraph(node.name, maxChars));
    const fontSize = 3;
    const borderRadius = 3;
    const padding = 10;

    ctx.font = `${fontSize}px Sans-Serif`;
    //creates the the square with the rounded corners
    const lines = label.split('\n');
    const lineHeight = fontSize * 1.4; // Adjust the line height as needed, 1.2 is for
        margin around it
    const maxLineWidth = Math.max(...lines.map(line => ctx.measureText(line).width)) +
        3;
    const bckgDimensions = [maxLineWidth, lines.length * lineHeight].map(n => n +
        fontSize * 0.2);

    ctx.fillStyle = mapNodeToColor(node, node.id);
    ctx.beginPath();
    ctx.moveTo(node.x - bckgDimensions[0] / 2 + borderRadius, node.y - bckgDimensions[1]
         / 2);
    ctx.lineTo(node.x + bckgDimensions[0] / 2 - borderRadius, node.y - bckgDimensions[1]
         / 2);
    ctx.quadraticCurveTo(node.x + bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2,
         node.x + bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2 + borderRadius);
    ctx.lineTo(node.x + bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2 -
        borderRadius);
    ctx.quadraticCurveTo(node.x + bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2,
         node.x + bckgDimensions[0] / 2 - borderRadius, node.y + bckgDimensions[1] / 2);
    ctx.lineTo(node.x - bckgDimensions[0] / 2 + borderRadius, node.y + bckgDimensions[1]
         / 2);
    ctx.quadraticCurveTo(node.x - bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2,
         node.x - bckgDimensions[0] / 2, node.y + bckgDimensions[1] / 2 - borderRadius);
    ctx.lineTo(node.x - bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2 +
        borderRadius);
    ctx.quadraticCurveTo(node.x - bckgDimensions[0] / 2, node.y - bckgDimensions[1] / 2,
         node.x - bckgDimensions[0] / 2 + borderRadius, node.y - bckgDimensions[1] / 2);
    ctx.closePath();
    ctx.fill();

    ctx.textAlign = 'center';
    ctx.textBaseline = 'middle';
    ctx.fillStyle = '#000000';

    const initialY = node.y - ((lines.length - 1) * lineHeight) / 2;

    lines.forEach((line, index) => {
      const y = initialY + index * lineHeight;
      ctx.fillText(line, node.x, y);
    });

    node.__bckgDimensions = bckgDimensions;
  }}

/>

// fgRef is used for manipulating the force paramter of the graph
const fgRef = useRef();

useEffect(() => {
const fg = fgRef.current;
fg.d3Force('link').distance(link => 20 * (1/(link.distance * link.distance)))
fg.d3Force('charge').strength(-20);
fg.d3Force('charge').distanceMax(100);
}, [root, threshold]);
```

Next to the changes to the forces within the graph, some additional functionality is added to the nodes of the complete graph. The graph view offers the option to *enable and disable navigation on click nodes*. If the option is enabled a user can click on a node in the complete graph to have one of the following happen:

- **For Guidelines:** If a selected node is a guideline the user is navigated to the guideline detail page.
- **For Categories:** If a selected node is a category or subcategory of the taxonomy then the complete graph for the children of the selected node is displayed, similarly selecting a category in the list representation of the taxonomy.

The user is given the option to enable and disable this navigation to prevent accidentally navigating to a new graph when clicking and dragging nodes in the force graph.

Collapsible hierarchical tree graph implementation While the collapsible tree graph is very close in functionality to the implementation of Srikugan's *VisGuideExplorer* [Sri21] the implementation makes use of the *react-d3-tree* library instead of directly implementing it using *D3.js*. The tree is implemented by creating a *renderNode* method, which renders nodes differently depending on whether they are the root node, an inner node or a leave node of the tree. The root node and inner nodes represent categories and subcategories of the taxonomy, while the leaf nodes are the specific guidelines. Selecting an inner node expands or collapses it while clicking on a leaf node navigates the user to the detailed guideline page.

Listing 4.7: Method to render nodes in the hierarchical tree graph

```
  /**
   * Renders the node differently depending on if it is a root node or not.
   * Additionally changes the color of a node if the node is expanded
   */
  function renderNode({ nodeDatum, toggleNode }) {
    const isLeaf = !nodeDatum.children || nodeDatum.children.length === 0;
    const isExpanded = nodeDatum.__rd3t.collapsed !== true;
    const currentClass = isExpanded ? "node__expanded" : "node";
    let guideline = '';

    if (isLeaf) {

      guideline = decodeUTF8String(nodeDatum.name)

      return (
        <Tooltip title="Click to navigate to details" placement="top-start">
          <g>
            <text
              ref={(el) => {
                if (el) {
                  const offsetX = 80;
                  el.setAttribute('x', offsetX);
                }
              }}
              fill="black"
              strokeWidth="0.5"
              onClick = { () => navigate(createLink(guideline, guidelines))}
            >
              {createLineBreaks("Guideline: " + guideline, 60)}
            </text>
            <circle
              cx="0"
              cy="0"
              className="node__leaf"
            />
          </g>
        </Tooltip>
      );
    } else {
      return (
        <Tooltip title={countChildren(nodeDatum.children)}>
          <g>
            <text
              ref={(el) => {
                if (el) {
                  const offsetX = calculateTextPosition(el);
                  el.setAttribute('x', offsetX);
                }
              }}
              fill="black"
              strokeWidth="1"
              onClick={toggleNode}
            >
              {nodeDatum.name}
            </text>
            <circle
              cx="0"
              cy="0"
              onClick={toggleNode}
              className={currentClass}
            />
          </g>
        </Tooltip>
      );
    }
  }
```

5 VisKnowsBest: Presentation and Evaluation

The following chapter presents the VisKnowsBest tool. First, the layout of the tool is presented and the basic navigation within the tool is explained. Then the content views or pages that provide the main functionality are introduced and the possible interactions with these pages is showcased.

5.1 Layout and navigation

The VisKnowsBest tool provides a simple layout, consisting of a header or navigation bar, a sidebar and content views [see Figure 5.1]. The header provides the main navigation functionality and a search bar for text input. The sidebar is conditionally rendered and provides functionality to adjust the results of the text search. Additionally, the sidebar provides functionality to adjust the presented related resources for the guidelines in the detailed guideline view. Finally, the main part of the layout is the content, which offers a variety of different views.

5.1.1 Navigation bar

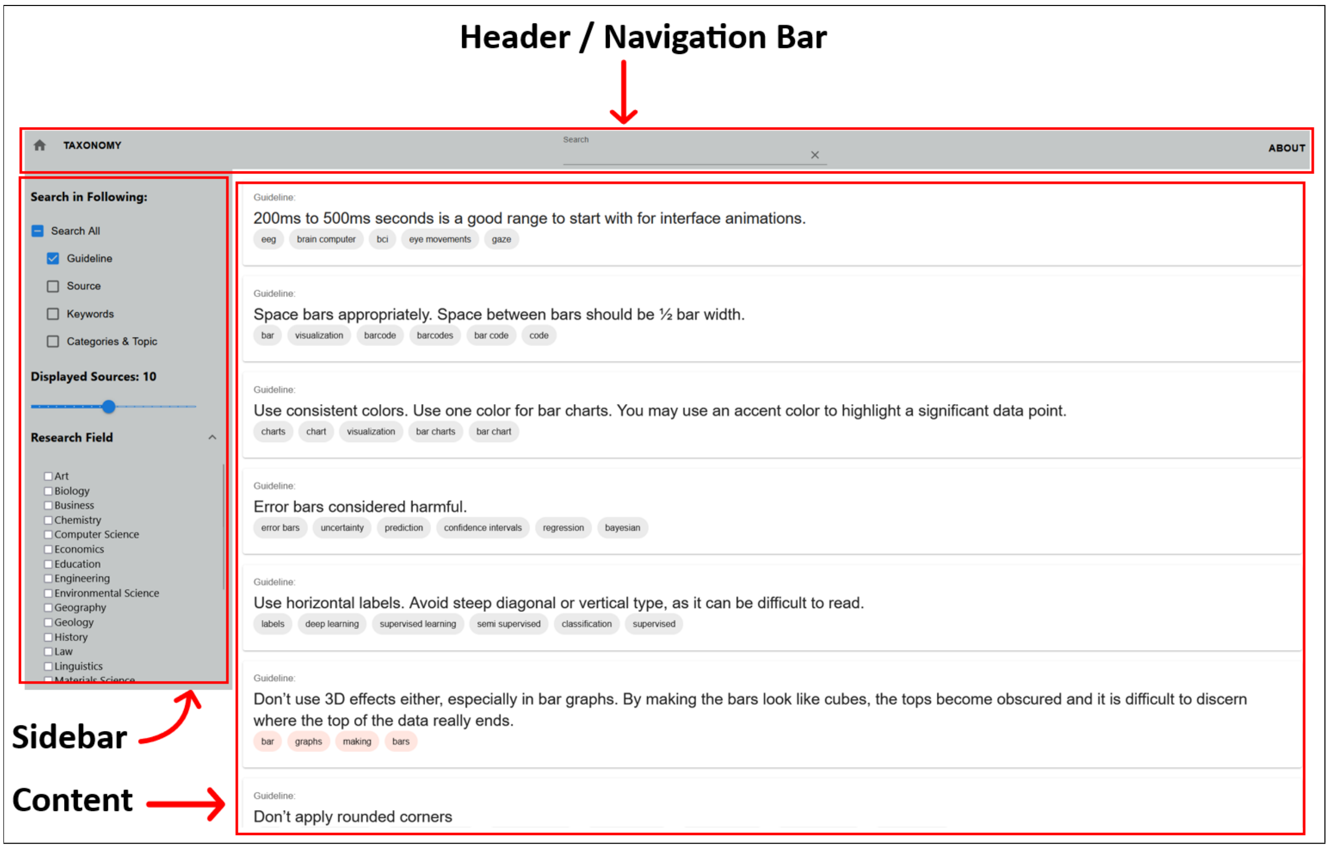
The header of the VisKnowsBest tool is the navigation bar [see Figure 5.2]. The navigation bar provides two main functionalities:

- **Navigation:** It offers navigation to four of the main content views of the tool. These four main content views are the *Guideline List* which also is the home screen of the tool, *Guideline Details Page*, *Taxonomy Views* and the *About Page*. The navigation functions by clicking on the desired view in the navigation bar. The different buttons responsible for the navigation provide additional information about their functionality to the user when hovering over them.
- **Search Bar:** The search bar allows for text input from the user and provides an easy way to search through all the guidelines present in the database. Depending on the content view, the search bar functions in two different ways: If the current content view is the *Guideline List* the displayed guidelines are filtered by the search input in real time and updated to only display guidelines fitting the search input. For any other content view, the search bar takes user input and upon pressing the *Enter Key* it navigates the user to the *Guideline List* and displays all guidelines fitting the search input of the user. The search bar makes use of Levenshtein distance when filtering the guidelines for the search input. This allows the user to receive search results even when having spelling errors in the input.

5.1.2 Sidebar

The sidebar [see Figure 5.3] provides functionality to adjust the search results of the search bar and offers the customization of the related resources that are provided for a guideline in the *Guideline Details Page*.

- **Adjusting Search Results:** The text search conducted with the search bar can be adjusted by changing what part of the guideline data is being searched. By default the text search searches in the guideline titles. Additionally, this search can also be changed to include the source of the guideline, the keywords related to the guideline and the guidelines categories and topics. When searching for the source this can include the author or the original publication, blog or resource the guideline was published in. The source field currently is limited to the data provided in the data set collected in [Sri21]. Any combination of adjustments of guideline title, source and keywords is possible with the exception of deselecting all of them.



Guideline:
200ms to 500ms seconds is a good range to start with for interface animations.
eeg · brain computer · bci · eye movements · gaze

Guideline:
Space bars appropriately. Space between bars should be ½ bar width.
bar · visualization · barcode · barcodes · bar code · code

Guideline:
Use consistent colors. Use one color for bar charts. You may use an accent color to highlight a significant data point.
charts · chart · visualization · bar charts · bar chart

Guideline:
Error bars considered harmful.
error bars · uncertainty · prediction · confidence intervals · regression · bayesian

Guideline:
Use horizontal labels. Avoid steep diagonal or vertical type, as it can be difficult to read.
labels · deep learning · supervised learning · semi supervised · classification · supervised

Guideline:
Don't use 3D effects either, especially in bar graphs. By making the bars look like cubes, the tops become obscured and it is difficult to discern where the top of the data really ends.
bar · graphs · making · bars

Guideline:
Don't apply rounded corners

Figure 5.1: Overview of the layout of the VisKnowsBest tool. The layout consists of three main components. The header or navigation bar on the top of the screen, a sidebar on the left side of the screen with the main content occupying the rest of the screen space.

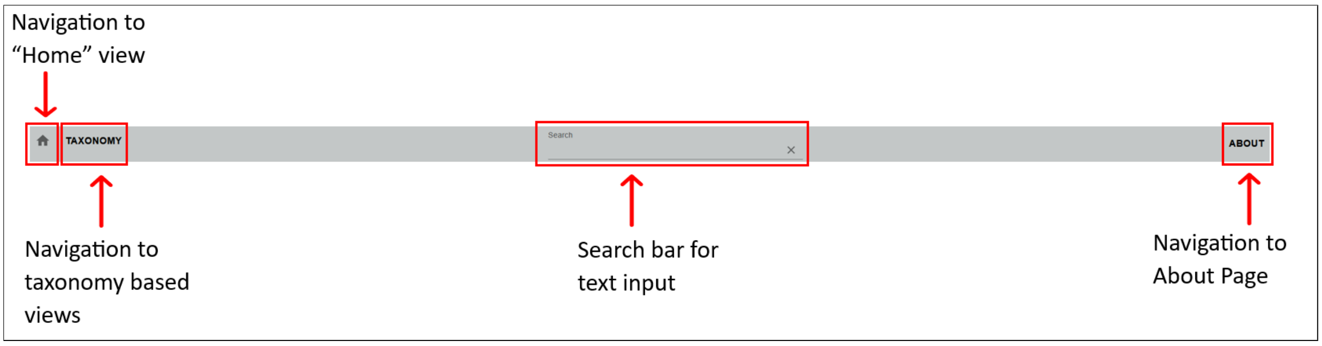
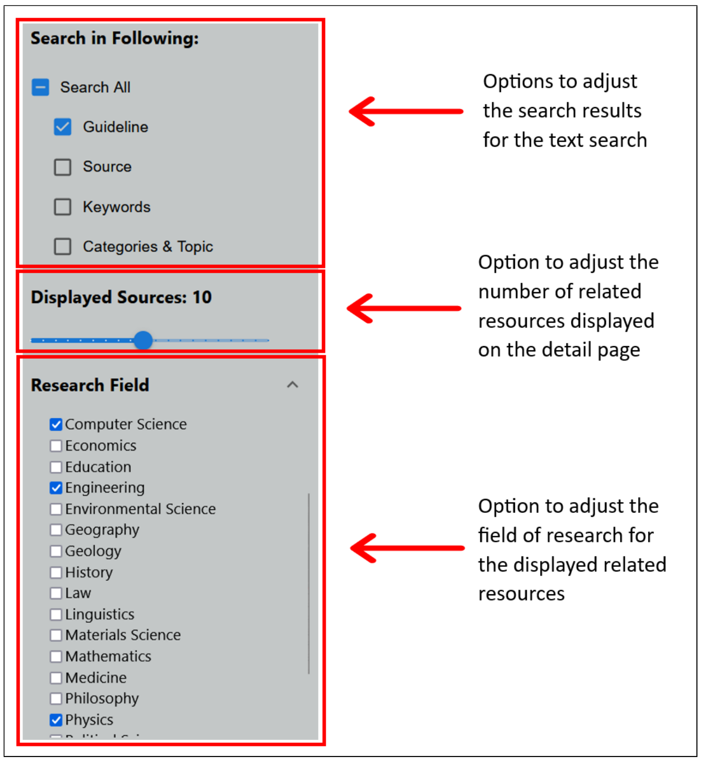

Navigation to "Home" view
TAXONOMY
Search
ABOUT
Navigation to taxonomy based views
Search bar for text input
Navigation to About Page

Figure 5.2: Navigation bar or header of the VisKnowsBest tool. Provides basic and intuitive navigation functionality. It can be used to navigate to the home tab (guideline list), to the taxonomy based views and to the *About* page. In the middle of the navigation bar, a search bar allows for text input from the user.



Search in Following:

- [✓] Search All
  - [✓] Guideline
  - [ ] Source
  - [ ] Keywords
  - [ ] Categories & Topic

Displayed Sources: 10

Research Field

- [✓] Computer Science
- [ ] Economics
- [ ] Education
- [✓] Engineering
- [ ] Environmental Science
- [ ] Geography
- [ ] Geology
- [ ] History
- [ ] Law
- [ ] Linguistics
- [ ] Materials Science
- [ ] Mathematics
- [ ] Medicine
- [ ] Philosophy
- [✓] Physics

Options to adjust the search results for the text search
Option to adjust the number of related resources displayed on the detail page
Option to adjust the field of research for the displayed related resources

Figure 5.3: Sidebar of the VisKnowsBest tool. The sidebar allows for adjusting the search conducted by the search bar by changing whether the search is for the guideline titles, sources and/or the keywords of the guidelines. Additionally, the sidebar allows the user to interactively change the displayed related resources of a guideline by changing the number of displayed resources and changing what scientific field the related resources are from. The current selection only displays ten related resources which are in the fields of *Computer Science, Engineering* or *Physics*.

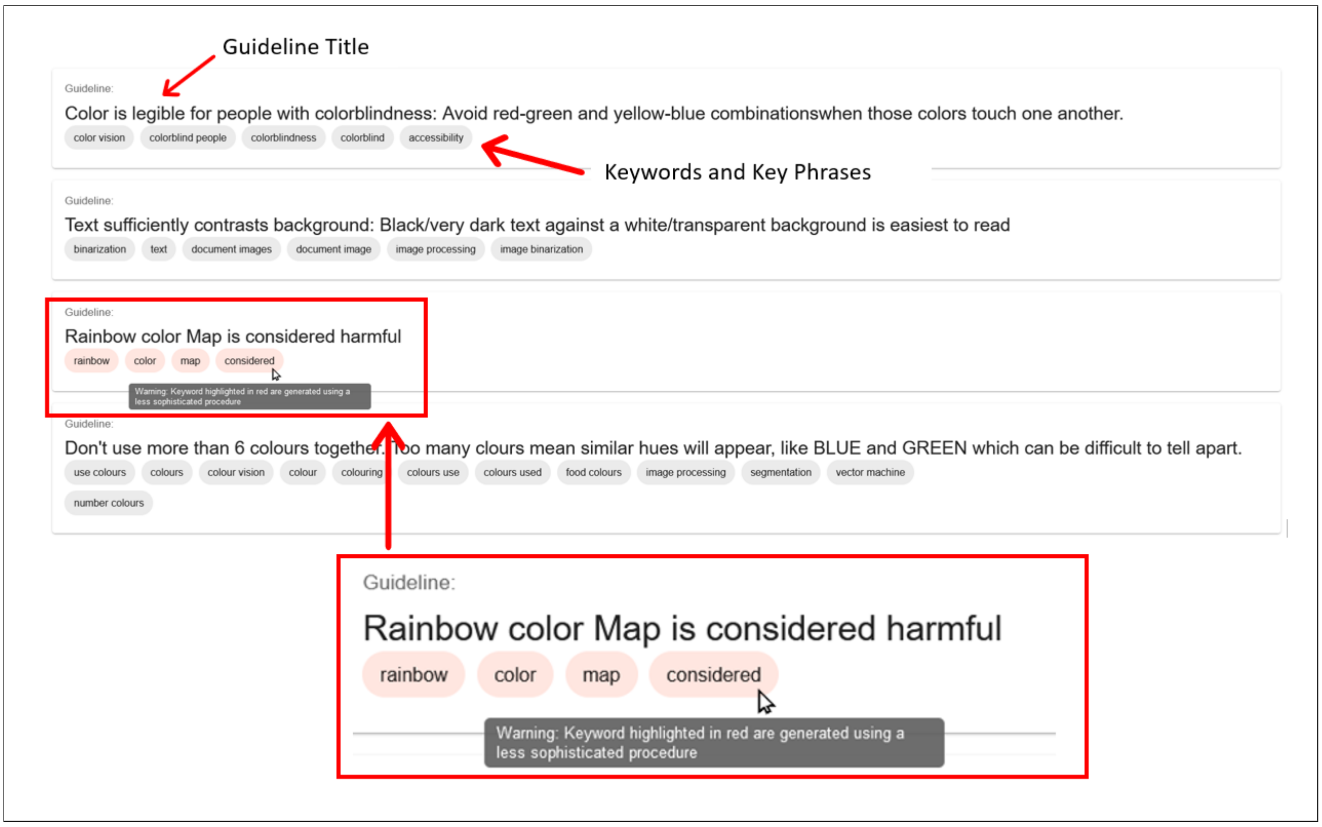
- **Customization Related Resources:** The displayed, related resources for a guideline in the *Guideline Details Page* can currently be interacted with in two ways. The number of displayed related resources can be changed to range anywhere from one to 20 resources. This is done by dragging the slide selector to the desired number of resources. The number of currently displayed resources is displayed above the slider. The second interaction allows the user to select which field of research the related resources have to be in. This can be done by selecting the fields of research in the list *Research Field*. The selection is inclusive, meaning if two or more fields are selected a resource can be from either one or multiple selected fields and it doesn't have to be in all the selected resource fields. If no field is selected then the research field isn't restricted, meaning publications from any field will be returned. The available options for fields of research are limited by the Semantic Scholar Academic Graph API, which is utilized to fetch and load the resources.

5.1.3 Content

The content views provide the main functionality, interaction and guidance to the user. The main content views include the *Guideline List*, which is also the content view displayed when entering the tool, three *Taxonomy Based Views*, a view displaying *Guideline Details* and an *About Page* which provides additional information for the tool.

5.2 Content views

The following section will present the different content views in detail. It will start off with the default view when entering the tool, the *Guideline List*, which provides a scrollable overview of the guidelines. Next, it will highlight



Guideline:
Color is legible for people with colorblindness: Avoid red-green and yellow-blue combinationswhen those colors touch one another.
color vision | colorblind people | colorblindness | colorblind | accessibility

Guideline:
Text sufficiently contrasts background: Black/very dark text against a white/transparent background is easiest to read
binarization | text | document images | document image | image processing | image binarization

Guideline:
Rainbow color Map is considered harmful
rainbow | color | map | considered
Warning: Keyword highlighted in red are generated using a less sophisticated procedure

Guideline:
Don't use more than 6 colours together. Too many clours mean similar hues will appear, like BLUE and GREEN which can be difficult to tell apart.
use colours | colours | colour vision | colour | colouring | colours use | colours used | food colours | image processing | segmentation | vector machine | number colours

Guideline:
Rainbow color Map is considered harmful
rainbow | color | map | considered
Warning: Keyword highlighted in red are generated using a less sophisticated procedure

Figure 5.4: The *Guideline List* displays guidelines as cards that contain an overview of the guidelines which includes the guideline title and the keywords related to the guideline. Zoomed in, a guideline card is displayed for a guideline where the keyword generation failed and therefore the seed keywords were used. These keywords are highlighted in red and a warning is displayed to the user when hovering over a keyword.

the *Guideline View*, which presents the user with details of a guideline on demand. In the next step, the three taxonomy-based views are presented. All of these views are based on a taxonomy used by Sajan Srikugan in his master's thesis [Sri21]. The first taxonomy view is a fully expanded tree graph, implemented as a force-directed graph, representing the taxonomy with all guidelines, categories and subcategories. The second taxonomy view is an explorable tree graph that allows for expanding and collapsing nodes and navigating to guidelines by selecting them. The third and final taxonomy-based view presents the tree structure of the taxonomy in a list-like structure while providing a complete graph of the direct children of the currently selected category or subcategory. The final content view presented in this section is the *About Page*.

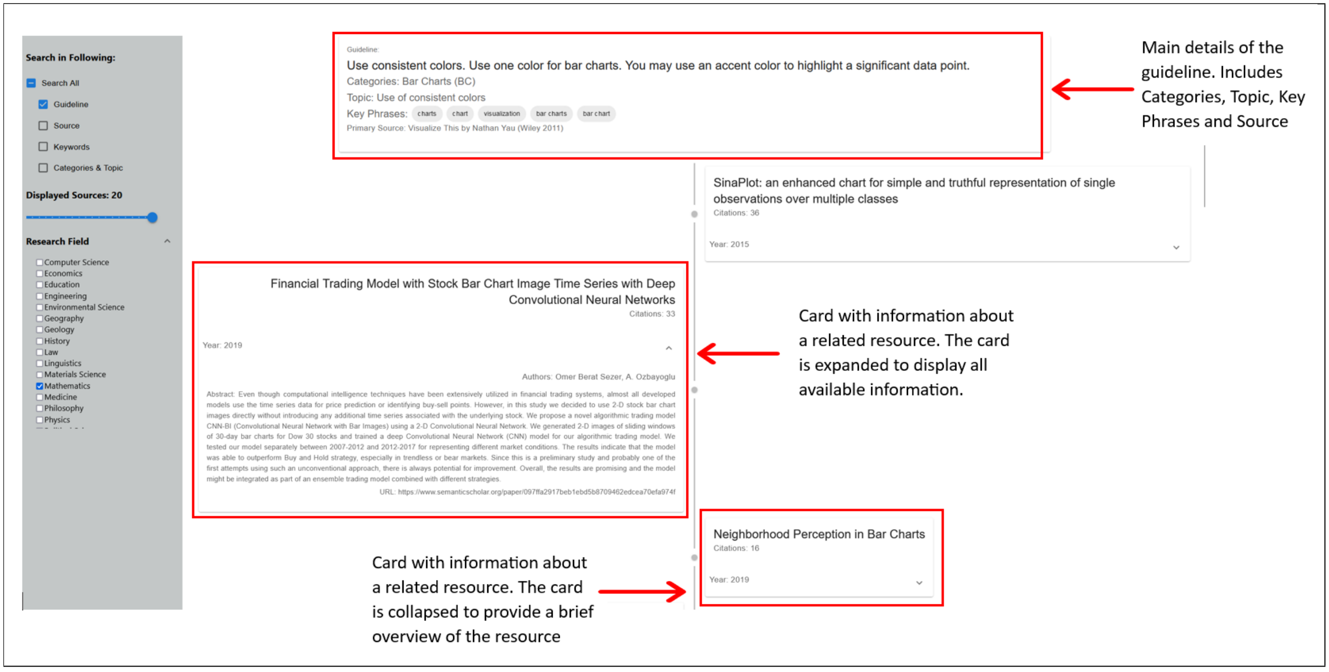
5.2.1 Guideline list

The *Guideline List* is the main content view of the tool. It provides a list, that can be searched by the user by providing text input in the search bar. The guidelines are updated in real-time to fit the search input. Additionally, the search input can be adjusted to search through the guideline title, sources, keywords and *topics & categories*. By clicking on any guideline a user is navigated to the *Guideline Details* of a guideline.

5.2.2 Guideline details

The *Guideline Details* view [see Figure 5.5] provides the user with details about any selected guideline. The details directly related to the guideline are displayed on a card and include:

- **Guideline Title:** This is the title of the guideline or just the guideline itself.



Search in Following:

- Search All
- [✓ Guideline]
- Source
- Keywords
- Categories & Topic

Displayed Sources: 20

Research Field

- Computer Science
- Economics
- Education
- Engineering
- Environmental Science
- Geography
- Geology
- History
- Law
- Linguistics
- Materials Science
- [✓ Mathematics]
- Medicine
- Philosophy
- Physics

Guideline:
Use consistent colors. Use one color for bar charts. You may use an accent color to highlight a significant data point.
Categories: Bar Charts (BC)
Topic: Use of consistent colors
Key Phrases: charts chart visualization bar charts bar chart
Primary Source: Visualize This by Nathan Yau (Wiley 2011)

Main details of the guideline. Includes Categories, Topic, Key Phrases and Source

SinaPlot: an enhanced chart for simple and truthful representation of single observations over multiple classes
Citations: 36
Year: 2015

Financial Trading Model with Stock Bar Chart Image Time Series with Deep Convolutional Neural Networks
Citations: 33
Year: 2019
Authors: Omer Berat Sezer, A. Ozbayoglu
Abstract: Even though computational intelligence techniques have been extensively utilized in financial trading systems, almost all developed models use the time series data for price prediction or identifying buy-sell points. However, in this study we decided to use 2-D stock bar chart images directly without introducing any additional time series associated with the underlying stock. We propose a novel algorithmic trading model CNN-BI (Convolutional Neural Network with Bar Images) using a 2-D Convolutional Neural Network. We generated 2-D images of sliding windows of 30-day bar charts for Dow 30 stocks and trained a deep Convolutional Neural Network (CNN) model for our algorithmic trading model. We tested our model separately between 2007-2012 and 2012-2017 for representing different market conditions. The results indicate that the model was able to outperform Buy and Hold strategy, especially in trendless or bear markets. Since this is a preliminary study and probably one of the first attempts using such an unconventional approach, there is always potential for improvement. Overall, the results are promising and the model might be integrated as part of an ensemble trading model combined with different strategies.
URL: https://www.semanticscholar.org/paper/097ffa2917beb1ebd5b8709462edcea70efa974f

Card with information about a related resource. The card is expanded to display all available information.

Neighborhood Perception in Bar Charts
Citations: 16
Year: 2019

Card with information about a related resource. The card is collapsed to provide a brief overview of the resource

Figure 5.5: Guideline Details Page for the guideline *Use consistent colors. Use one color for bar charts. You may use an accent color to highlight a significant data point.*. Currently, 20 related resources are loaded and the resources are selected to be in the field of research of *Mathematics*. The resource *Financial Trading Model with Stock Bar Chart Image Time Series with Deep Convolutional Neural Networks* is expanded to display the author, abstract and URL.

- **Categories:** Categories and subcategories assigned to the guideline in the context of the taxonomy used in the VisKnowsBest tool.
- **Topic:** (Optional) The topic that has been assigned to a guideline. Only available for guidelines that have a topic assigned to them
- **Key Phrases:** Key phrases and keywords related to the guideline. The keywords and -phrases offer additional search functionality. By clicking on a keyword, the user is navigated to the guideline list, which is filtered to only display guidelines that have the selected keyword or -phrase assigned to them.
- **Sources:** Source of where the guideline has been extracted from. Sources are additionally annotated to display whether they are primary, secondary or tertiary sources.

Displaying this information should provide a user with additional guidance with respect to the guideline while introducing as little bias as possible. The categories and topics aim to provide the user with an orientation about the applicability of the guideline. The keywords and key phrases can provide additional context to the guideline while also aiding searchability. Additionally, the keywords and -phrases are used in fetching related resources for the guidelines which in turn aim to provide further context for the guidelines.

The related resources of a guideline are displayed underneath the main guideline card, displaying all the details. The related resources are sorted in decreasing order by the number of their citations and display the name of the publication, number of citations and the year of the publication by default. When expanded the related resources additionally display authors, an abstract and an URL to access the complete resource. By allowing the user to interact with and manipulate the displayed resources, the fourth step of designing guidance is taken into account and therefore allows the user to adapt the guidance to their specific needs [CAA$^+$20].

5.3 Taxonomy based views

The VisKnowsBest tool includes three taxonomy-based views. It makes use of a taxonomy used and applied by Sajan Srikugan in his master thesis [Sri21]. The different taxonomy views aim to put the guidelines and their categorization into context for the user and therefore provide additional guidance. The first taxonomy-based view displays the complete taxonomy, including all guidelines, categories and subcategories. It is presented in a tree structure as a force-directed graph. The second taxonomy-based view is a tree graph that allows the user to explore the taxonomy by expanding and closing nodes, which represent categories. The leaf nodes of the graph represent the guidelines. The final taxonomy-based view contains the taxonomy in a list-like form on the left-hand side of the screen and a complete graph of all children of the currently selected category. The complete graph is a force-directed graph where the links are adjusted in their width and length to visualize the semantic similarity of the two connected guidelines.

5.3.1 Force graph of complete taxonomy

This taxonomy-based view provides the user with an overview of the complete taxonomy [see Figure 5.6]. Due to the number of guidelines, subcategories and categories, this graph is very cluttered and overwhelming. While the graph allows only very limited insight with respect to specific guidelines it aims to provide the user with context about the categories and subcategories within the taxonomy. The force graph works by assigning each link and node an equal value, while the links pull the nodes together, the nodes repulse each other. Because each node has the same repulsion factor, each branch of the tree creates a cluster, as parents and children are attracted to each other by the links between them, while different branches push each other away. If a branch is richly populated, meaning it has many subcategories and guidelines belonging to it, it will create a cluster that strongly repulses any other branch. This results in sparsely populated branches being close to the centre of the graph while richly populated branches position themselves further away from the centre. This allows the user to gain an overview of how richly populated certain categories are by seeing the size and the positioning of the clusters.

5.3.2 Hierarchical topic tree of taxonomy

This hierarchical topic tree of the taxonomy allows the user to interactively explore the taxonomy [see Figure 5.7]. This representation of the taxonomy is heavily based on an implementation by Srikugan which was created during a taxonomic study of visualization guidelines [Sri21]. The hierarchical topic tree allows the user to expand and collapse nodes by clicking on them. When a user hovers over a node the number of direct children of that node also is displayed to them. This more interactive and explorable representation aims to provide a more structured understanding of the taxonomy and the positioning of the guidelines within that taxonomy to the user. The leaf nodes of this graph represent the guidelines and allow a user to navigate to the guidelines detail page by clicking on them, adding another avenue of navigation.

5.3.3 Complete graph and expandable taxonomy list

This view showcases the taxonomy on the left side of the screen in the form of an expandable and collapsible list. The rest of the screen presents a complete graph as a force-directed graph [see Figure 5.8]. Upon first accessing the view the nodes of the initial complete graph include all main categories of the taxonomy. By either navigating through the list representation of the taxonomy or by selecting any node on the complete graph the view is updated to display the complete graph of all direct children of the selected node. The user has the option to deactivate the navigation upon clicking on one of the nodes of the force-directed graph to simplify the rearranging of the nodes within the graph and to avoid accidentally navigating to a subgraph by clicking instead of dragging a node.

The complete graph showcases the relationships between categories or guidelines by assigning the link between more similar nodes a wider width and a shorter distance, while dissimilar nodes are connected by a longer and thinner link.

Further, the complete graph can be adjusted by only displaying links that represent connections between nodes with similarities above a certain threshold. This can be modified by the user by adjusting the threshold using a

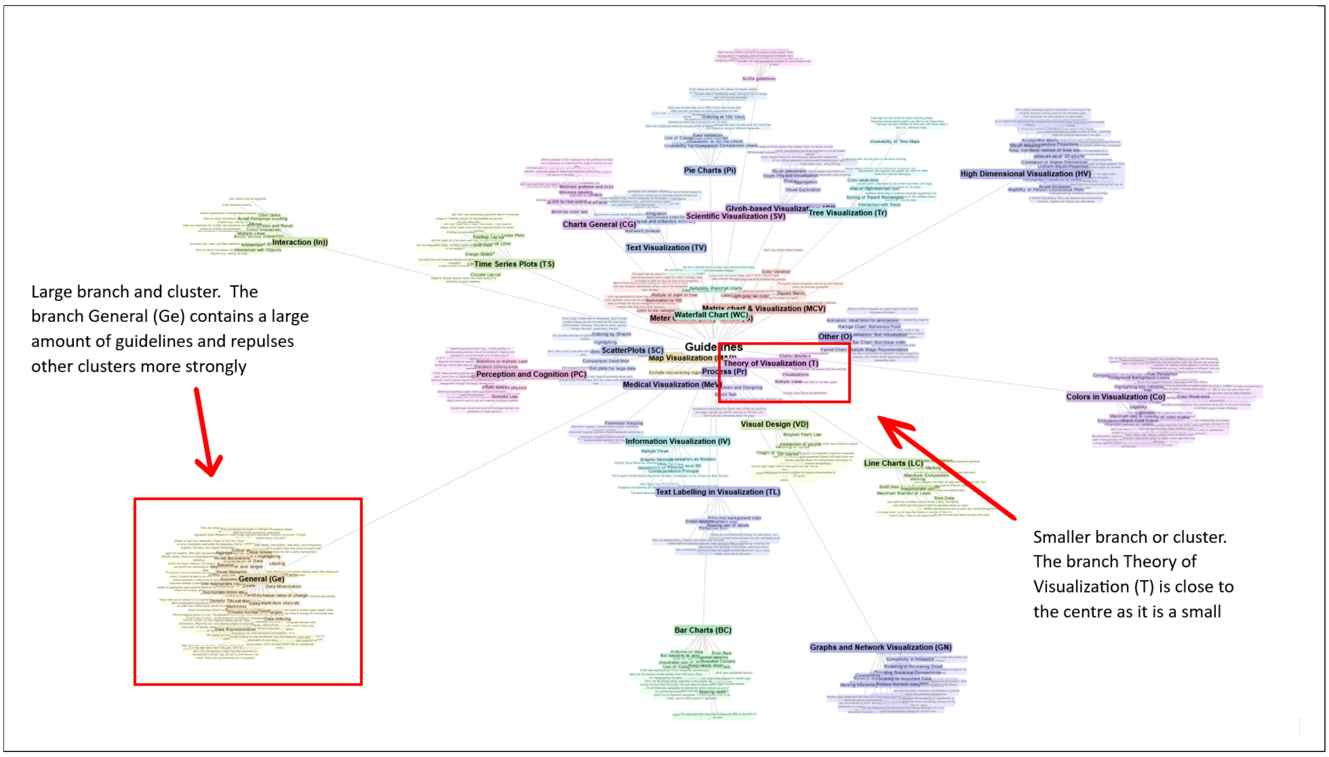

Large branch and cluster. The branch General (Ge) contains a large amount of guidelines and repulses other clusters more strongly
Smaller branch or cluster. The branch Theory of Visualization (T) is close to the centre as it is a small
Guidelines
Theory of Visualization (T)
General (Ge)
Bar Charts (BC)
Graphs and Network Visualization (GN)
Colors in Visualization (Co)
High Dimensional Visualization (HV)
Line Charts (LC)
Visual Design (VD)
Text Labelling in Visualization (TL)
Information Visualization (IV)
Perception and Cognition (PC)
Medical Visualization (MeV)
Interaction (In)
Time Series Plots (TS)
Charts General (CG)
Text Visualization (TV)
Pie Charts (Pi)
Scientific Visualization (SV)
Tree Visualization (Tr)
Matrix chart & Visualization (MCV)
Waterfall Chart (WC)
ScatterPlots (SC)
Map Visualization
Process (Pr)
Other (O)

Figure 5.6: Force Graph representing the structure of the taxonomy in a tree-like structure. For categories and subcategories with more children, there are larger clusters and therefore the repulsion from the centre is larger. The closer to the centre a category is the more sparsely it is currently populated.

slider ranging from zero to one. A link with a value of zero represents no semantic similarity between two nodes, while a link with a value of one means the two nodes it connects are semantically identical.

5.3.4 About page

The About page is a view that provides the user with additional information about the VisKnowsBest tool. This includes information about the data collection and generation process, a list of used technologies and background information on the taxonomy used in this thesis. By providing users with this additional information it offers up transparency and builds trustworthiness. Additionally, it allows users to reason about the provided data by understanding how it got collected and what exactly the keywords represent.

5.4 Evaluation of the requirements

This section evaluates the requirements defined in the preceding chapter. The functional and non-functional requirements as well as any additional requirements will be evaluated on whether they have been met or not. It will also explain, how the requirements were met or why they were not met. Requirements that were added in the course of the thesis to expand the base functionality will be marked with the word *Additional*.

Functional requirements

- **A scrollable list representing an overview of all guidelines should be visible on the home screen:** This requirement is satisfied with the implementation of the guideline list, which provides an overview of all guidelines as a scrollable list.

- **Selecting a guideline in the guideline list navigates to the guidelines details page where additional information (author, origin, additional resources) for the guideline is displayed:** This requirement is

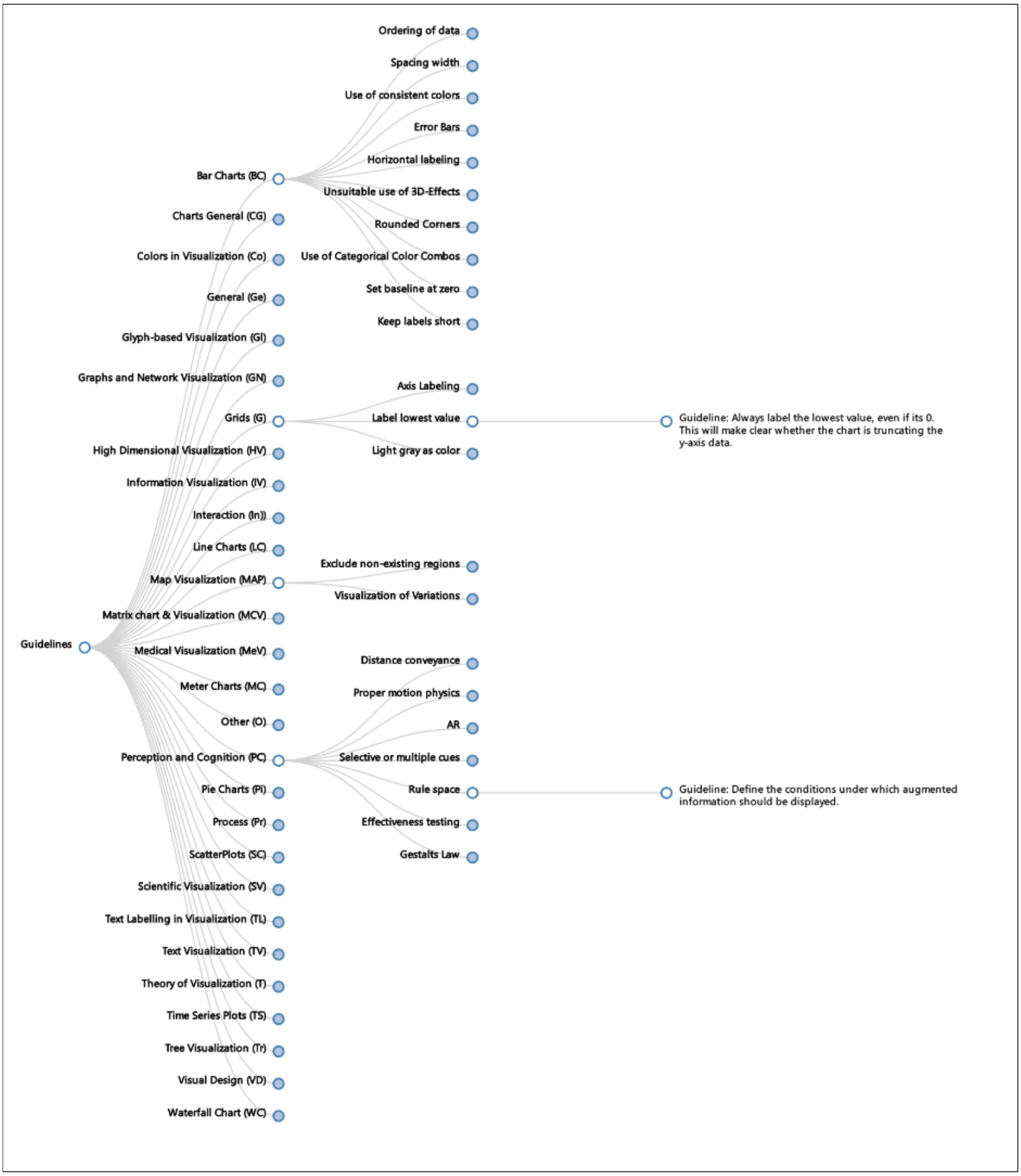
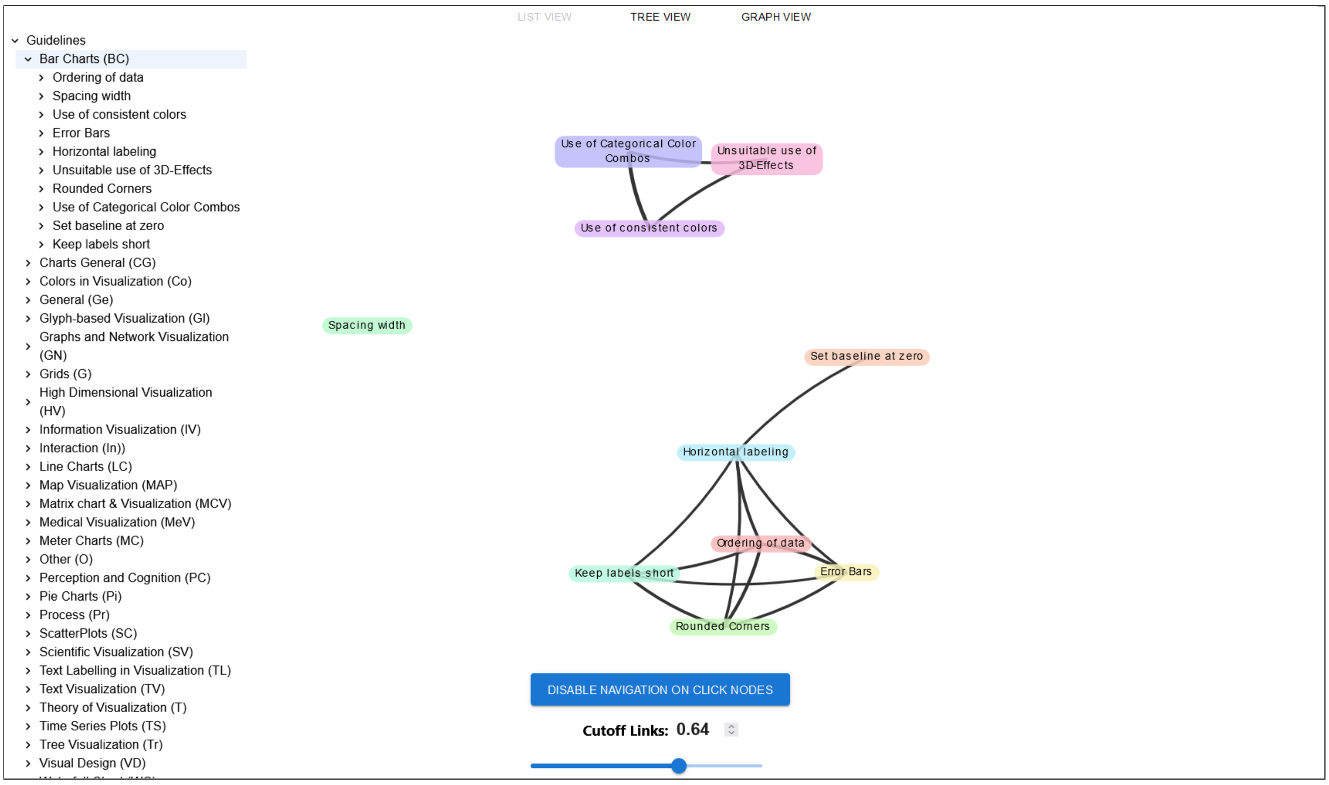

Guidelines
Bar Charts (BC)
Charts General (CG)
Colors in Visualization (Co)
General (Ge)
Glyph-based Visualization (Gl)
Graphs and Network Visualization (GN)
Grids (G)
High Dimensional Visualization (HV)
Information Visualization (IV)
Interaction (In))
Line Charts (LC)
Map Visualization (MAP)
Matrix chart & Visualization (MCV)
Medical Visualization (MeV)
Meter Charts (MC)
Other (O)
Perception and Cognition (PC)
Pie Charts (Pi)
Process (Pr)
ScatterPlots (SC)
Scientific Visualization (SV)
Text Labelling in Visualization (TL)
Text Visualization (TV)
Theory of Visualization (T)
Time Series Plots (TS)
Tree Visualization (Tr)
Visual Design (VD)
Waterfall Chart (WC)
Ordering of data
Spacing width
Use of consistent colors
Error Bars
Horizontal labeling
Unsuitable use of 3D-Effects
Rounded Corners
Use of Categorical Color Combos
Set baseline at zero
Keep labels short
Axis Labeling
Label lowest value
Light gray as color
Exclude non-existing regions
Visualization of Variations
Distance conveyance
Proper motion physics
AR
Selective or multiple cues
Rule space
Effectiveness testing
Gestalts Law
Guideline: Always label the lowest value, even if its 0. This will make clear whether the chart is truncating the y-axis data.
Guideline: Define the conditions under which augmented information should be displayed.

Figure 5.7: Partially expanded Hierarchical Tree Graph of the Taxonomy. The categories *Bar Charts (BC), Grids (G), Map Visualization (MAP)* and *Perception and Cognition (PC)* are expanded. The subcategories *Label lowest value* and *Rule space* are further expanded to show their corresponding guidelines.



- Guidelines
  - Bar Charts (BC)
    - Ordering of data
    - Spacing width
    - Use of consistent colors
    - Error Bars
    - Horizontal labeling
    - Unsuitable use of 3D-Effects
    - Rounded Corners
    - Use of Categorical Color Combos
    - Set baseline at zero
    - Keep labels short
  - Charts General (CG)
  - Colors in Visualization (Co)
  - General (Ge)
  - Glyph-based Visualization (GI)
  - Graphs and Network Visualization (GN)
  - Grids (G)
  - High Dimensional Visualization (HV)
  - Information Visualization (IV)
  - Interaction (In))
  - Line Charts (LC)
  - Map Visualization (MAP)
  - Matrix chart & Visualization (MCV)
  - Medical Visualization (MeV)
  - Meter Charts (MC)
  - Other (O)
  - Perception and Cognition (PC)
  - Pie Charts (Pi)
  - Process (Pr)
  - ScatterPlots (SC)
  - Scientific Visualization (SV)
  - Text Labelling in Visualization (TL)
  - Text Visualization (TV)
  - Theory of Visualization (T)
  - Time Series Plots (TS)
  - Tree Visualization (Tr)
  - Visual Design (VD)

Figure 5.8: Taxonomy in the shape of a collapsible list. Next to the taxonomy the complete graph of all children of *Bar Charts(BC)* is displayed. All links with a similarity rating lower than 0.64 are cut off, according to user preference, resulting in 3 separate clusters.

addressed and met with the guideline details page. The page can be reached by selecting a guideline in the list and additionally by selecting a guideline in two of the three taxonomy-based views. The details page itself provides the required information and additional information to the scope of this requirement

- **A collapsible and expandable hierarchical tree graph representing the taxonomy allows users to explore the taxonomy interactively:** This requirement is addressed and met with the hierarchical tree graph implementation of the taxonomy based view. Additionally, the implementation allows the user to navigate to the guideline details view. To also view and explore the whole expanded taxonomy additionally, the complete taxonomy as a force-directed graph was added to provide a different visualization of the taxonomy.

- **A search bar allows users to conduct text searches for the guidelines:** This requirement is met and addressed by the search bar in the VisKnowsBest header. The addition of allowing for small typing errors in the search input via a Levenshtein distance further improves the implementation of this requirement.

- **(Additional) Options for the user to adjust the search results to search by source, categories and keywords should be available to allow users to tailor the text search to their needs:** This requirement is addressed by the option to adjust search results in the sidebar of the VisKnowsBest tool. Meeting this requirement comes with the slight downside of slightly longer load times if searching within all fields of a guideline.

- **(Additional) Options for the user to adjust the additional resources provided by the details view of a guideline should be available to adjust the resources to their needs:** This requirement is addressed and met by the functionality to adjust the number of resources and their field of research in the sidebar. While this requirement is met, additional filtering options for a user could further improve the quality of this functionality.

- **(Additional) A graph representing the semantic similarity of categories and guidelines within the chosen taxonomy should be implemented to provide a user with additional context about the taxonomy:** The taxonomy list and complete graph view address and meets this requirement to a limited degree. While the complete graph represents the similarity between categories or guidelines by adjusting the links width and length approximately a more sophisticated calculation mechanism is needed to represent these values more accurately.

Non-functional requirements

- **The UI is clean and clutter free :** This requirement is met by creating a clear and consistent layout for the VisKnowsBest tool. With the header, sidebar and content views there are only three major components to the UI. To avoid unnecessary complexity and clutter the sidebar is not rendered for any of the taxonomy-based views, as its functionality only applies to the guideline list and guideline details page.

- **The UI provides simple and intuitive navigation for a user:** This requirement is addressed by positioning all the major navigation options in the header bar. Additionally, the users have different options to navigate to the guidelines details page by selecting the guideline in the guideline list or by clicking on a guideline in one of the taxonomy-based views. By providing these natural and intuitive navigation options this requirement is met.

- **All guidelines together with their provenance data should be accessible to the user on selection with loading times being under one second:** The loading times for the application are mainly under one second. The one exception to this is the search functionality. When a user selects to search in all fields the wait time can be one to three seconds. This is due to the additionally added Levenshtein distance calculation. Therefore this requirement is only partially met. The removal of the Levenshtein distance would ensure that this requirement is met, but the gained quality of the Levenshtein-aided search vastly outweighs this slight drawback, which only applies under very specific conditions.

Overall all the main requirements are met by the VisKnowsBest tool. The only base requirement not being met is the loading time being under one second, which under very specific circumstances can exceed one second. As this is due to the implementation of an additional quality-of-life feature that outweighs the impact of the waiting times, the decision was made to accept not fulfilling this requirement. In addition, to the base requirements, some more requirements were added and fulfilled, adding additional functionality and quality-of-life improvements for the users.

6 Conclusion and Discussion

The field of visualization and the *theory of visualization* provide a wide range of opportunities for research and contributions. This thesis focuses on its contribution to the field of visualization guidelines. The implementation of the VisKnowsBest tool offers possible solutions for prominent problems in the world of visualization guidelines. The tool provides solution approaches for the main problem of the distributed nature of visualization guidelines on the internet and in any other form of media. Further, the VisKnowsBest tool tackles the problem of trustworthiness and transparency of visualization guidelines by providing their sources and additional resources for the user to enable them to understand the context of a visualization guideline. The VisKnowsBest tool provides a prototype for a scalable web repository with the potential of providing a framework for a central source of knowledge for visualization guidelines. By providing an intuitive, uncluttered, clean and searchable interface, the tool aims to provide high accessibility to the world of visualization guidelines. By allowing users to search the guidelines using generic text input, selecting tags in the form of keywords or exploring a hierarchical tree diagram based on a taxonomy, VisKnowsBest provides different means for searching and exploring the space of visualization guidelines, fitting for different users and tasks. Allowing users to modify the text search further allows for an interactive search, adjusted to the users' preferences. All of this allows the tool to be usable and useful for a wide range of users such as researchers, practitioners and students.

The tool aims to provide guidance with learning intent by letting the user explore, search and even manipulate related resources which provide additional context to the guidelines. By presenting the guidelines in the context of a taxonomy, VisKnowsBest also offers guidance with a teaching intent by providing a categorization of the guidelines and comparing guidelines depending on their semantic similarity. Implying this relationship between guidelines based on their categorization and quantifying it by a value based on the semantic similarity of their related keywords allows users to reason on relationships between guidelines and their assigned category in the context of the chosen taxonomy.

Next to the tool itself, this thesis provides different approaches to provide additional data for visualization guidelines. A semi-automated keyword generation process allows for inputting a single guideline and generating a set of keywords and key phrases by conducting a keyword extraction based on related publications provided by the semantic scholar API. Generated keywords and other data get cleaned and loaded into the database of the VisKnowsBest tool to make it accessible for any user. By providing additional related resources, the user can interactively fetch user-specific context and guidance, while keeping the user-side bias to a minimum.

6.1 Limitations

One of the main limitations of this work is the data provided and presented in the VisKnowsBest tool. While the quantity of the data exceeds the goal of 5-15 guidelines, the quality of the generation process for the related keywords and -phrases could still be augmented. This is due to the shift from following a largely manual work-intensive approach over to the finally used mainly automated approach. The more automated approach highlights the value and potential of an automated pipeline for the generation of data related to the guidelines. The downside of the automated approach is, that it falls short of the scrutiny and therefore quality a manual approach offers. An additional shortcoming related to the keywords is the fact that the keyword generation process fails for a small subset of guidelines, which makes it necessary to use a simplified keyword generation process for these guidelines.

A further limitation is based on the use of the current taxonomy, which is already highlighted in Srikugans work. The taxonomy uses a naive approach which is illustrated for instance in the fact that each guideline only is assigned one category, while a guideline can have relevance in more than one of the categorizations. Additionally to this, the general use of a taxonomy can be criticised as there currently is not a universally accepted taxonomy in the field of visualization guidelines and therefore the choice of any taxonomy introduces a certain bias towards the chosen taxonomy.

The focus of the project was on the implementation of a visual exploration tool and interactive features, not in the collection of guidelines, one of the major limitations of the tool still is the large number of guidelines currently not available in the VisKnowsBest data base.

The missing ability to directly interact with fellow researchers and practitioners via a discussion feature in the VisKnowsBest tool or linking related discussions on discussion platforms such as VisGuides is a further limitation. Additional to this interaction between users of the tool, any feature to edit, add, annotate or delete information from the guidelines has been omitted. While the code provides the infrastructure for this, the tool itself does not allow a user to add, change or delete data on the database, as this would require a quality control process to be implemented to ensure the quality of the provided data.

A final quality-based limitation of the VisKnowsBest tool is the lack of an automated unit- and integration test suite and lacking security features in the code, which is attributed to the limited time frame and the experimental, prototype-like nature of the provided tool. To ensure safe hosting and a bug-free experience in an online environment, this is an essential quality deficit of the current code.

6.2 Future Work

On the technical side, code quality can be improved by addressing quality issues related to security and testing in the code. Once these issues are solved and the infrastructure is available, the tool can be hosted and made accessible for any user. Further quality and efficiency improvements to the search functionality will become necessary if there is a significant increase in provided data. The quality and quantity of the provided guidelines can be improved, for instance by implementing a semi-automated process for adding, curating and updating guidelines. Such an approach could be based on the supply chain based ViSupply model [EARC18], to ensure consistency within the provided data and crucially also a necessary level of scrutiny and scientific rigour. Adding on to this, the linking of related discussions on the VisGuides page or similar discussion forums would provide a significant quality improvement to the site. Currently, the lack of a publicly accessible API limits the possibility of this functionality.

Research on taxonomies and ontologies for guidelines as well as any approaches of structuring and categorizing guidelines provide further potential for the VisKnowsBest tool. Adding new and existing taxonomies as well as updating and correcting currently used taxonomies have the potential to increase the quality of information and guidance provided to users of the tool.

Bibliography

[ACD⁺] Mohammad Alharbi, Jian Chen, Alexandra Diehl, Dylan Rees, Elif E Firat, Qiru Wang, and Robert S Laramee. Visualization resources. `https://sites.google.com/view/visres/home`. Accessed: 2023-03-15.

[Ast] Vasco Asturiano. react-force-graph. `https://github.com/vasturiano/react-force-graph`. Accessed: 2023-06-28.

[BKC⁺13] Rita Borgo, Johannes Kehrer, David Chung, Eamonn Maguire, Robert Laramee, Helwig Hauser, Matthew Ward, and Min Chen. Glyph-based visualization: Foundations, design guidelines, techniques and applications. *Eurographics State of the Art Reports*, pages 39–63, May 2013.

[BTI07] David Borland and Russell Taylor Ii. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, 27(2):14–17, 2007.

[CAA⁺20] Davide Ceneda, Natalia Andrienko, Gennady Andrienko, Theresia Gschwandtner, Silvia Miksch, Nikolaus Piccolotto, Tobias Schreck, Marc Streit, Josef Suschnigg, and Christian Tominski. Guide me in analysis: A framework for guidance designers. *Computer Graphics Forum*, 39(6):269–288, 2020.

[Cam23] Cambridge dictionary - guidance. `https://dictionary.cambridge.org/dictionary/english/guidance`, March 2023. Accessed: 2023-03-27.

[Ces19] Amy Cesal. Datavisualizationstyleguiderepositories. `https:https://www.datavisualizationsociety.org/resources`, June 2019. Accessed: 2023-03-13,.

[CGJ⁺17] Min Chen, Georges Grinstein, Chris Johnson, Jessie Kennedy, and Melanie Tory. Pathways for theoretical advances in visualization. *IEEE Computer Graphics and Applications*, 37(4):103–112, 2017.

[CGM⁺17] Davide Ceneda, Theresia Gschwandtner, Thorsten May, Silvia Miksch, Hans-Jörg Schulz, Marc Streit, and Christian Tominski. Characterizing guidance in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):111–120, 2017.

[CGM19] Davide Ceneda, Theresia Gschwandtner, and Silvia Miksch. A review of guidance approaches in visual data analysis: A multifocal perspective. *Computer Graphics Forum*, 38(3):861–879, 2019.

[Coc05] Alistair Cockburn. Hexagonal architecture. `https://alistair.cockburn.us/hexagonal-architecture/`, 2005. Accessed: 2023-05-29.

[COSK21] Jinhan Choi, Changhoon Oh, Bongwon Suh, and Nam Wook Kim. Toward a Unified Framework for Visualization Design Guidelines. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7. ACM, May 2021.

[DARB⁺22] Alexandra Diehl, Alfie Abdul-Rahman, Benjamin Bach, Mennatallah El-Assady, Matthias Kraus, Robert Laramee, and Min Chen. Characterizing grounded theory approaches in visualization. March 2022.

[DAREA$^+$] Alexandra Diehl, Alfie Abdul-Rahman, Mennatallah El-Assady, Benjamin Bach, Daniel Keim, and Min Chen. Visguides. `https://visguides.org/`. Accessed: 2023-03-08.

[DAREA$^+$18] Alexandra Diehl, Alfie Abdul-Rahman, Mennatallah El-Assady, Benjamin Bach, Daniel Keim, and Min Chen. Visguides: A forum for discussing visualization guidelines. In *Eurographics Conference on Visualization*, 2018.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, May 2019.

[DKAR$^+$20] Alexandra Diehl, Matthias Kraus, Alfie Abdul-Rahman, Mennatallah El-Assady, Benjamin Bach, Robert Steven Laramee, Daniel A. Keim, and Min Chen. Studying Visualization Guidelines According to Grounded Theory. *ArXiv*, abs/2010.09040, October 2020.

[EARC18] Ulrich Engelke, Alfie Abdul-Rahman, and Min Chen. Visupply: A supply-chain process model for visualization guidelines. In *2018 International Symposium on Big Data Visual and Immersive Analytics (BDVA)*, pages 1–9, 2018.

[Few11] Stephen Few. The Chartjunk Debate – a close examination of recent findings, 2011.

[Few12] Stephen Few. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press, 2nd ed. edition, 2012.

[Fin] Financial times - visual vocabulary. `https://ft-interactive.github.io/visual-vocabulary/`. Accessed: 2023-03-08.

[Flo] Flowingdata, data visualization and statistics. `https://flowingdata.com/`. Accessed: 2023-03-08.

[FWD$^+$17] Hui Fang, Simon Walton, Emily Delahaye, James Harris, Dmitry Storchak, and Min. Chen. Categorical colormap optimization with visualization case studies. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):871–880, 2017.

[Gro20] Maarten Grootendorst. Keyword extraction with bert. `https://www.maartengrootendorst.com/blog/keybert/`, October 2020. Accessed:2023-06-12.

[Gro21] Maarten Grootendorst. Keybert. `https://zenodo.org/record/4461265`, January 2021. Accessed:2023-06-12.

[IBM] What is data visualization? `https://www.ibm.com/topics/data-visualization`. Accessed: 2023-03-08.

[Inf] Information is beautiful. `https://informationisbeautiful.net/`. Accessed: 2023-03-08.

[Joh04] Chris Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, 2004.

[KG] Jeff Kunkle and Tobias Gurtzick. db-migrate. `https://db-migrate.readthedocs.io/en/latest/`. Accessed: 2023-06-28.

[Kir] Andy Kirk. Visualising data. `https://www.visualisingdata.com`. Accessed: 2023-03-08.

[KL16] Eser Kandogan and Hanseung Lee. A Grounded Theory study on the language of data visualization principles and guidelines. *Electronic Imaging*, 28:1–9, February 2016.

[Kre] Ben Kremer. react-d3-tree. `https://github.com/bkrem/react-d3-tree`. Accessed: 2023-06-28.

[KW11] Christa Kelleher and Thorsten Wagener. Ten guidelines for effective data visualization in scientific publications. *Environmental Modelling & Software*, 26(6):822–827, June 2011.

[LAC+23] Xiaoxiao Liu, Mohammad S Alharbi, Jian Chen, Alexandra Diehl, Dylan Rees, Elif E Firat, Qiru Wang, and Robert s Laramee. Visualization resources: A survey. *Information Visualization*, 22(1):3–30, 2023.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[NJG21] Vinh Nguyen, Kwanghee Jung, and Vibhuti Gupta. Examining data visualization pitfalls in scientific publications. *Vis. Comput. Ind. Biomed. Art*, 4:27, December 2021.

[Oxf23] Oed oxford english dictionary - guideline. `https://www.oed.com/view/Entry/89823771?redirectedFrom=guideline#eid`, March 2023. Accessed: 2023-03-07.

[PMCEA+22] Ignacio Pérez-Messina, Davide Ceneda, Mennatallah El-Assady, Silvia Miksch, and Fabian Sperrle. A typology of guidance tasks in mixed-initiative visual analytics environments. *Computer Graphics Forum*, 41(3):465–476, 2022.

[Quo] Quora. `https://www.quora.com/`. Accessed: 2023-03-03.

[r/d] r/dataisbeautiful. `https://www.reddit.com/r/dataisbeautiful/`. Accessed: 2023-03-03.

[Red] Reddit. `https://www.reddit.com/`. Accessed: 2023-03-02.

[RT98] Bernice. Rogowitz and Lloyd Treinish. Data visualization: the end of the rainbow. *IEEE Spectrum*, 35(12):52–59, 1998.

[r/v] r/visualization. `https://www.reddit.com/r/visualization/`. Accessed: 2023-03-03.

[SB18] James Scott-Brown. Presenting visualization guideline collections. *2nd IEEE VIS 2018 Workshop on the Creation, Curation, Critique and Conditioning of Principles and Guidelines in Visualization (2018)*, 2018.

[SCEA23] Fabian Sperrle, Davide Ceneda, and Mennatallah El-Assady. Lotse: A practical framework for guidance in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 29:1124–1134, 2023.

[Shn96] Ben Shneiderman. The eyes have it: A task by data type taxonomy for. In *Proceedings of IEEE Symposium on Visual Languages*, volume 96, pages 336–343, 1996.

[SJB+20] Fabian Sperrle, Astrik Jeitler, Jürgen Bernard, Daniel A. Keim, and Mennatallah El-Assady. Learning and teaching in co-adaptive guidance for mixed-initiative visual analytics. In *EuroVis Workshop on Visual Analytics (EuroVA)*, 2020.

[SL19] Prafull Sharma and Yingbo Li. Self-supervised contextual keyword and keyphrase retrieval with self-labelling, August 2019.

[SMWH17] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017.

[spa] spacy. `https://spacy.io/`. Accessed: 2023-06-13.

[Sri21] Srikugan Srikugan. Vis guides explorer - a comprehensive and semi-automatic study of visualization guidelines, February 2021.

[SSM11] Samuel Silva, Beatriz Santos, and Joaquim Madeira. Using color in visualization: A survey. *Computers & Graphics*, 35:320–333, 04 2011.

[SSMT13] Hans-Jörg Schulz, Marc Streit, Thorsten May, and Christian Tominski. Poster: Towards a characterization of guidance in visualization, 01 2013.

[Sta] Stackoverflow. `https://stackoverflow.com/`. Accessed: 2023-03-03.

[TM04] Melanie Tory and Torsten Möller. Rethinking visualization: A high-level taxonomy. In *IEEE Symposium on Information Visualization*, pages 151–158, 01 2004.

[Tuf01] Edward Tufte. The visual display of quantitative information. In *The Visual Display of Quantitative Information*. Graphics Press, 2001. Originally published in 1983.

[vW06] Jarke Jan van Wijk. Views on visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):421–432, 2006.