Christoph Söls, BSc

Eye Tracking Support for Interactive and Adaptive Visual Exploration

MASTER'S THESIS
to achieve the university degree of
Diplom-Ingenieur

submitted to
Graz University of Technology

Supervisors
Univ.-Prof. Dipl.-Volksw. Dr. rer. nat. Tobias Schreck, M.Sc.
Dipl.-Ing. Dr.techn. Stefan Lengauer, BSc BSc
Institute of Visual Computing

Graz, April 2025

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material that has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present thesis.

Date

Signature

Acknowledgements

I would like to thank Tobias and Stefan for providing nothing else than excellent guidance from start to finish of this work. Acknowledgments also go to my Mother, Father, Brother and Christina, who went above and beyond to support me through my whole study time. This cannot be taken for granted. Thank you.

Graz, April 2025
Christoph

Abstract

Eye Tracking (ET) is a key technique for collecting user data during interactive visual exploration. The A$^+$CHIS document exploration system, which is a prototype application for an adaptive and interactive Consumer Health Information System (CHIS) and offers a variety of texts and visualizations with varying difficulties, however, lacks this technology. In this work, we propose a system that is able to track a user's gaze point, extract relevant information from the user interface, and structure it. By applying this information in form of a summarizable timeline (timeline analysis), hierarchic knowledge visualization (summative analysis), as well as component- and content based recommendations, we show multiple possibilities for making use of the collected data. The results from our evaluation and user-study prove that the system is capable of providing valuable functionality regarding interface adaptation and user information to A$^+$CHIS. Furthermore, the implementation can be extended and used to employ adaptability and explainability in web applications.

Contents

CHAPTER 1

Introduction

This chapter offers a description of the task at hand, the reason for engaging with it and a summary of the remaining chapters of this Master's thesis.

1.1 Challenges for User Tracking in Visual Exploration

Adaptive visualization lives from handling user-data gathered during visual exploration. While traditional methods of logging user input like mouse- or keyboard-interactions are precise and well established, ET, with increasing accuracy over the years, offers a new dimension for collecting interactions between user and system [1] and is able to provide benefits to the overall visual exploration process of a user [2].
The A$^+$CHIS[1] document exploration system, which is further described in Chapter 2, investigates possibilities in the field of adaptive visual exploration in the medical domain by conducting studies and implementing findings into a prototype application. Although it features manual user feedback and mouse tracking, ET, a commonly used mechanic to measure user interest [3], offers substantial support.

1.2 Our Design for Eye Tracking and Applications in a CHIS

The aim of this work is threefold: (1) Connecting an eye tracker to an existing visual exploration system and thus establishing support for future research; (2) providing tracked user interactions in a structured manner, and lastly (3) using this data to offer meaningful information about viewed content with three usage examples. In the first usage example whe showcase a user's exploration history as a timeline of (grouped) fixations, to obtain an overview of the viewed content. This also relates to the topic of explainability (Section 2.4),

[1] https://apchis.cgv.tugraz.at/

as users can retrace why certain recommendations were made. The second usage example is showcasing a knowledge collection based on an interactive hierarchy of barcharts to filter (grouped) fixations, while also offering a comparison between fixation length/count. The third usage example of the data is a content- and component-based recommender system using features of the nearest neighbors (components) and Tf-idf (content) (see Section 3.4.3).
This basic framework allows not only to adapt visualizations in a web-based document exploration system, but also to be extended and used in studies to investigate user behavior, as described in Section 6.

1.3 Outline of remaining Chapters

Chapter 2 introduces similar works in the field of ET in adaptive visual exploration and we discuss relevant assumptions. Chapter 3 is about the concepts and theoretical background necessary for the implementation, which is shown in Chapter 4, together with hard- and software specifications. In Chapter 5, we present the evaluation methodology, show and discuss results and give a prospect for future work. Lastly, Chapter 6 gives an overview about the core outcomes of the work and concluding remarks are made.

CHAPTER 2

Related Work

This chapter gives an overview of research findings in similar areas and topics and shows the connection to the concepts of this work.

2.1 The A$^+$CHIS Document Exploration System

The A$^+$CHIS document exploration system [4], which is under development during the creation of this thesis, is an FWF[1]-funded project, developed in collaboration with the Medical University of Graz[2] and the University of Graz[3]. The main objective of this project is to explore possibilities for user adaptation in a CHIS, relying on evidence-based medical documents.
The continuously developed web-application offers a variety of texts and visualizations with different levels of complexity. Figure 2.1 shows a screenshot of one of the exploration pages featured in A$^+$CHIS. Section 3.3 offers a detailed description of the relevant components for this work.

2.2 Eye tracking in User Adaptive Visual Exploration

Jianu et al. [5] propose a set of design considerations regarding the implementation of ET into an adaptive visual exploration system, which are shown in Figure 2.2. They present modalities for the data source, adaptation and usage, as well as possible limitation factors for them.

Generally, there are two modes of applying ET in the context of adaptive visual exploration, according to Duchowski et al. [6]. First, the *interactive* mode, where the gaze point is used as an alternative input method. Examples are listed by Oyekoya and Stentiford [1]. In early stages

[1] https://www.fwf.ac.at/
[2] https://www.medunigraz.at/
[3] https://www.uni-graz.at/

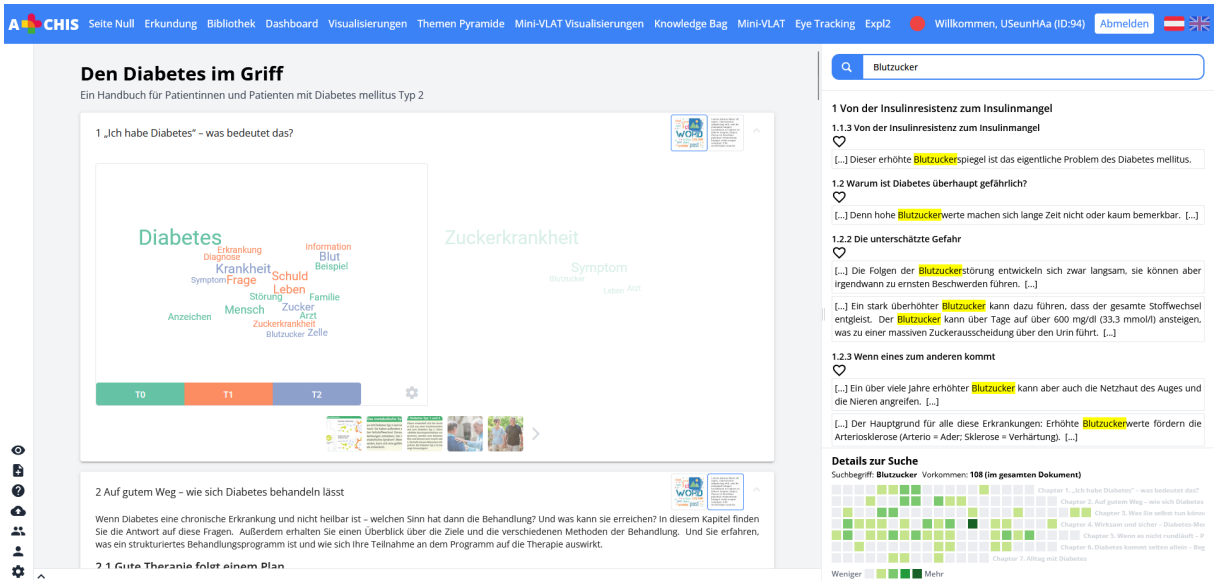


Figure 2.1: The *Exploration*-page of the A$^+$CHIS prototype.

of development of A$^+$CHIS, a Bachelor's thesis has been conducted by Lukasser [7] (using a different ET device than in this thesis), which came to the conclusion, that the approach is applicable to the system, although it suffers from inaccuracies. Second, the *diagnostic* mode, where various measures such as fixation time or saccade length are computed and evaluated. This work is focusing on the latter.

Conati et al. [8] state, that user adaptation by analyzing eye-gaze data is possible for various tasks, including both, adaptation regarding input and navigation methods, and simplification of graphical representations. Steichen et al. [9] show this by conducting a user study, where a simple graph is compared to a more complicated one. Conati et al. [8] summarize the conclusions of this and other studies, and found, that users with low perceptual speed work better with bar-charts, while users with high visual working memory prefer radar-plots. These findings can help people with different cognitive abilities get the most information out of visualizations, based on their needs.

Regarding interestingness (in our case applied for user recommendations and graphical presentation of information), Harsh et al. [10] state, that time spent in a Region of Interest (ROI) as well as the number of fixations may indicate a high interest, but it could also mean confusion for the current viewer. They also found, that the expertise level of viewers could be determined by investigating these metrics based on the current ROI. Also, according to Henderson and Hollingworth [11], the length of individual fixation points could indicate confusion, and the total number of fixation points/ total duration spent in an area of interest demonstrates the anticipated significance. Xu et al. [12] successfully implement such a recommendation system, which relies on the attention time on a specific object. They report

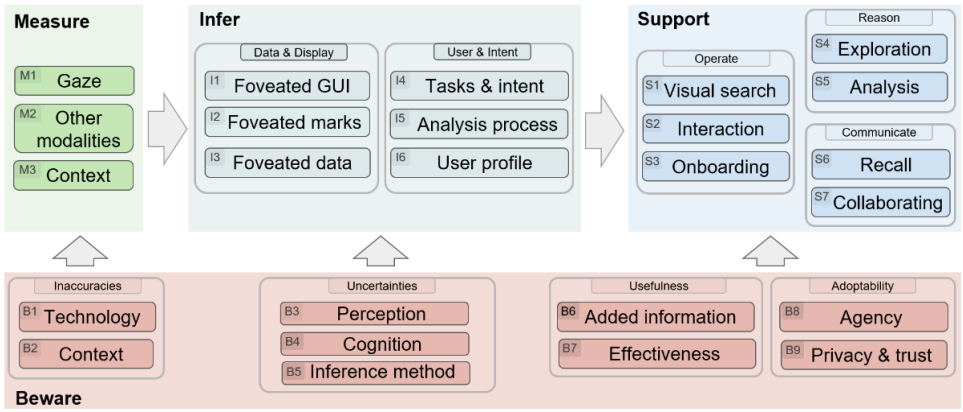


Figure 2.2: The elements for designing gaze-aware visualizations with data provided by the ET-device (Measure), additionally computable data (Infer), applications (Support) and limitations (Beware) [5].

more agreeable results compared to current benchmark systems.

Visual confusion, which is a very interesting topic regarding recommendations for adaptive systems, is a highly discussed field. It could help adapting complicated graphical representations to easier ones. This is especially important for users with low visualization literacy [13],[14], as a wider range of people get to understand provided information. Pachman et al. [15], who conduct a thinking aloud study in the domain of digital learning, do not find correlation between fixations and confusion when considering relevant areas, but positive (reported) confusion when looking at non-relevant ones. DeLucia et al. [16] and Graesser et al. [17], on the other hand, concluded, that there is a functional dependence between (total) fixation time and confusion. However, Graesser et al. [17] did not define precise metrics for confusion and DeLucia et al. [16] focus on the whole screen and not on a specified ROI.

2.3 Information Graphs

Knowledge graphs are mostly represented in a node-link style, where the edges define a relationship between two nodes, according to Hogan et al. [18]. For displaying hierarchical data, Shneiderman [19] proposes the tree-map, where size comparisons are intuitively possible. Especially for viewing times of similar ROIs, this is interesting, as users could retrace their previous steps. Compared to bar charts, the comparison between non-leaf nodes is easier possible. On the other hand, bar charts are better able to show the comparison between leaf nodes, especially at smaller datasets, according to Kong et al. [20]. A comparison between barcharts and treemaps is shown in Figure 2.3.

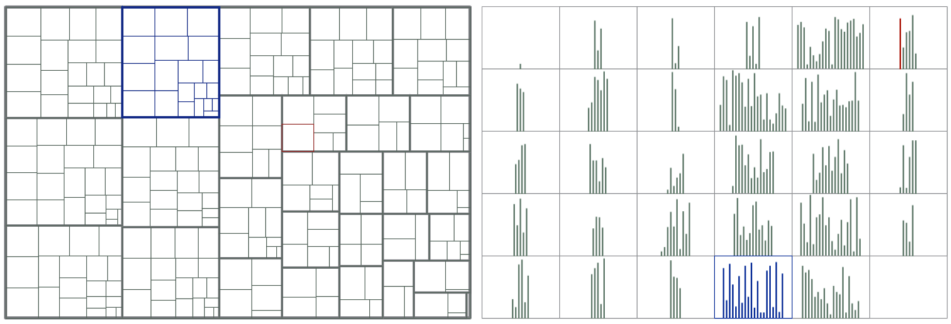


Figure 2.3: A comparison between a treemap and a bar chart showing the same data [20].

2.4 Explainability

Nowadays, when Large Language Models (LLMs) and Artificial Intelligence (AI) are as prevalent as never before, the interest in the topic of explainability and the demand for integration in applications is very high, according to Weber et al. [21]. The general aim of Explainable Artificial Intelligence (XAI), is to offer meaningful information about a model, to, among other things, gain trustworthiness among humans, as stated by Ali et al. [22]. One area of the explanation process is data explainability, where summarized data is presented and and can be analyzed by the user, to gain insight and understand part of the recommendation process. In our case, this will be implemented via history/timeline visualizations.

2.5 User Recommendations

By updating the interface with items a user finds interesting, adaptive exploration systems are part recommender systems. However, when analyzing fixations and saccades with statistical measures only, it is hard to infer a user interest due to the wide range of visual perception among humans, as described by Oyekoya and Stentiford [1]. By filtering out the necessary texts and images a user has looked at, this task becomes an example for content-based recommendation, as our current system does not provide user management for a collaborative structure. Lops et al. [23] provide an example architecture of such a system, as shown in Figure 2.4.

This approach combines similar texts with user ratings to suggest interesting content. This method has an essential drawback: Over-specialization. By always recommending similar texts, a user falls into a “loop” and rarely finds something completely new. Furthermore, recommendation systems may also strengthen certain biases, such as confirmation bias. Schwind and Buder [24] state, that providing *preference inconsistent* recommendations reduce said bias, at least in user groups with little prior domain knowledge.

According to Schreck et al. [4], recommendation algorithms used in CHISs mostly integrate

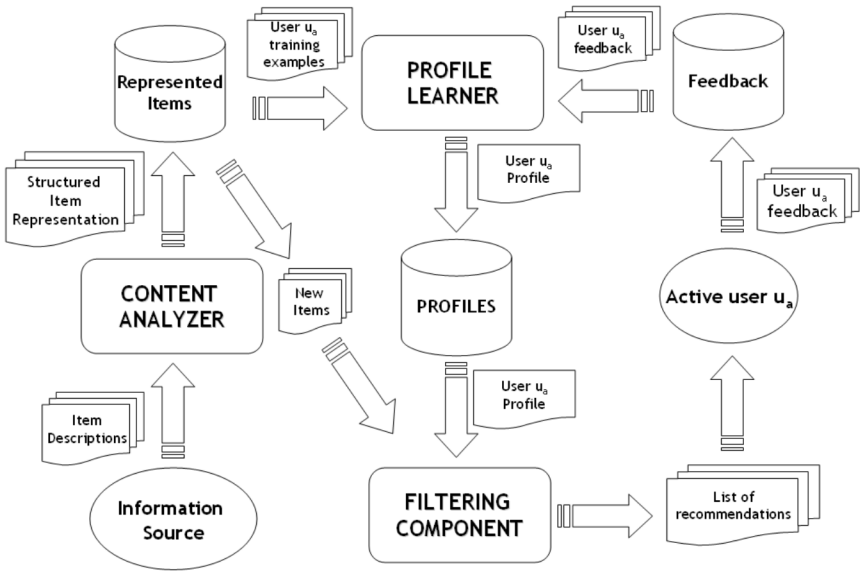


Figure 2.4: An example structure for a content-based recommending system [23].

only one dimension of representation, e.g. textual *or* graphic components. However, medical data often comes in the form of heterogenous data.

CHAPTER 3

Concepts

This chapter describes the ideas and the corresponding theoretical background for the implementations of this work. It includes assumptions and explanations for functions and metrics which are later used in Chapter 4, Implementation and Chapter 5, Evaluation.

3.1 Eye Tracking in Web-Applications

Most scientific papers in the ET-domain analyze user behaviors in a closed-off study environment. However, the foundation of this work is to offer a framework to an already fully functional web-application. The basic architecture for ET-implementations varies strongly and is usually defined by the used ET-device.

Similar to the concept by Jianu et al. [5], we introduce a dimensionality model to describe the ET-process. This way, we can visualize a range of differing components and usage scenarios with varying data offers and needs. As shown in Figure 3.1, we differentiate between three dimensions. The *Component* dimension contains all interesting visualizations to be tracked. Our main focus in this work lies on the *Fulltext*, *WordCloud* and *Nutri Pyramid* components, which are explained in Section 3.3. The second dimension, *What to track?*, lists available data which is collected by the ET-device directly, or extracted from the application depending on the gaze position. The *Usage* dimension connects the data with our usage scenarios, which are introduced in Section 3.4. This model can be adapted and extended for future work and similar applications.

3.2 Fixation Definition and Calculation

As pointed out by Carter et al. [25], to make results in the field of ET-research comparable, they need to show the methods for computing and describing fixations. Fixations are focus

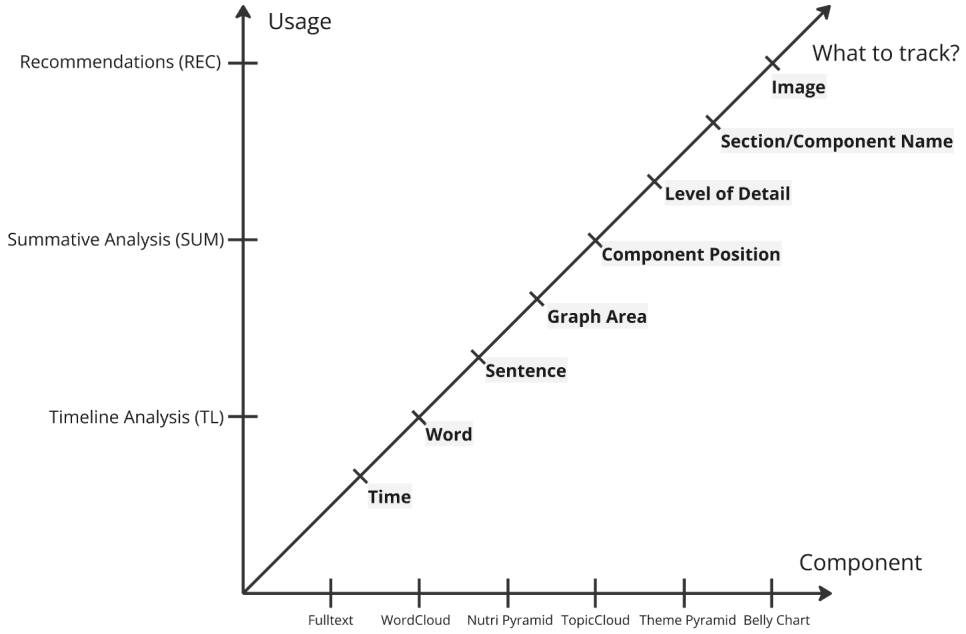


Figure 3.1: Our dimensionality model with components, tracked metrics and the inferred usage.

points where the eyes remain for a small period of time, to acquire information. Saccades, which are in between fixations, are movements where we cannot collect any information. An illustration of fixations and saccades while reading a text (word-based) is shown in Figure 3.2.

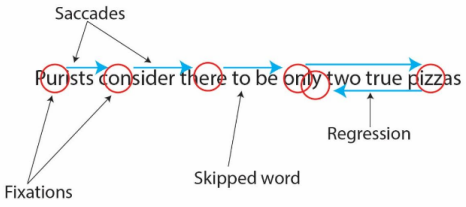


Figure 3.2: An illustration of fixations and saccades [25].

For calculating fixations, Salvucci and Goldberg [26] have analyzed a selection of prominent examples of algorithms. A table with their findings is shown in Figure 3.3. The selection of Velocity-Threshold Identification (I-VT) as the main algorithm for fixation detection in our work has following reasons: (1) the ease of implementation and (2) the fact, that *tobii*, which is the manufacturer of our ET-device (see Section 4.1), describe the same method being used in their own proprietary software, as explained by Olsen [27]. They also describe noise reduction by using a low-pass filter and gap-filling, both of which we use for preparing the raw gaze data offered by the eye-tracker.

Method	Accuracy	Speed	Robustness	Impl. Ease	Parameters
Velocity Threshold (I-VT)	√	√√	×	√√	1
Hidden Markov Model (I-HMM)	√√	√	√√	√/×[a]	8/0[a]
Dispersion Threshold (I-DT)	√√	√	√√	√	2
Minimum Spanning Tree (I-MST)	√	×	√√	×	2
Area-of-Interest (I-AOI)	×	√	√	√	1+[b]

Key: √√ = very good, √ = good, × = not as good.

Figure 3.3: A comparison between different fixation-detection algorithms [26].

The I-VT algorithm gets the already preprocessed gaze coordinates and labels each point based on the velocity relative to the previous coordinate. Based on the results of Olsen and Matos [28], as well as testing with different thresholds, the velocity value for a fixation classification was set to 30 degrees per second (°/s). To get the resulting fixation, the centroid of the classified fixation and saccade coordinates is computed. Implementation details are further described in Chapter 4.

3.3 Component Tracking

The following subsections describe the modifications the template documents have to undergo, before they can be tracked successfully. Furthermore, the first dimension - *Components* is introduced along with the 3 main information representations this work is focusing on.

General DOM Modifications

Generally, the template of a web application is built with the principles of the Document Object Model (DOM), which is the connection between the HTML language and the JavaScript/TypeScript logic and represented as a tree structure. To track this structure solely by screen coordinates, we propose a method, where the whole tree is tracked with a fixed sampling rate of 10 Hz, and filtered based on pre-set tags on relevant nodes in the DOM. These tags are divided into 4 levels by information type and position, and also include meta information about the node, as shown in Table 3.1 and Figure 3.4.

Table 3.1: DOM-tag classes.

Level	Description
0	*Text/Position.* Word, sentence or graph position.
1	*Component.* Graph, text or other Area of Interest (AOI).
2	*Modal/Popup* (optional)
3	*Section.* Part of the application the user is currently navigating in.

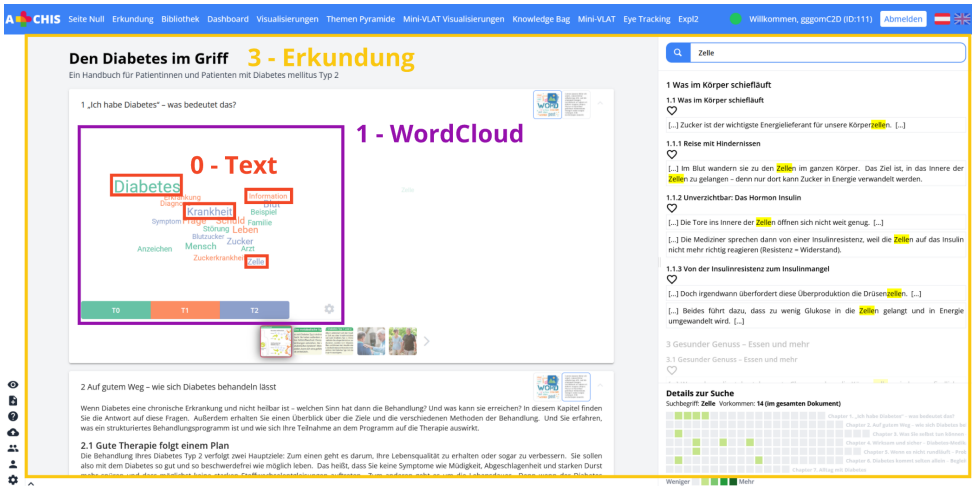


Figure 3.4: The DOM-tag classes with corresponding elements.

Because of the wide range of information representations our document exploration system offers, we are facing a problem in level 0: The detection of the atomic information unit we can obtain from a representation, how we generalize, and how we represent it. The next sections will address this problem.

Navigation

Besides the 3 main components of interest, which have undergone further customizations, the majority of pages and elements in the DOM have been tagged, as described in Paragraph 3.3. These components, which include control- and selection elements such as buttons or drop-downs, will be labeled as "Navigation" - in contrast to the main components, basically being fixed AOIs, which are described as "Focus". The tracked metrics of these components include Time, Component Position, Section/Component Name and Image.

3.3.1 Texts

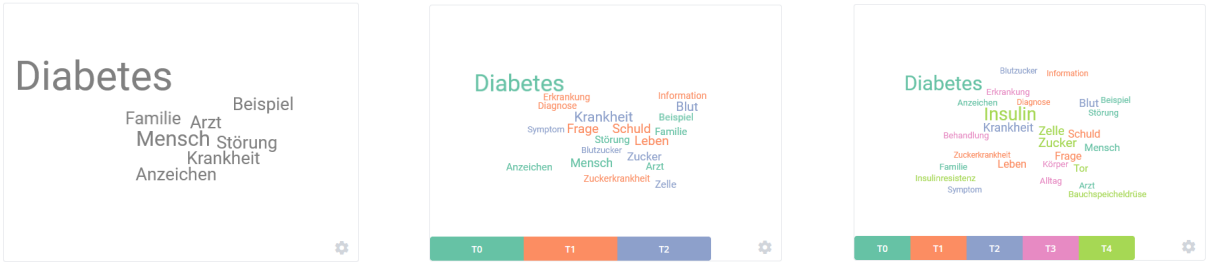
The third considered component is the *Fulltext* component, which displays the textual contents of a PDF, that were previously extracted to a sentence-based data structure. Thus, the smallest piece of information is a sentence, or - in case of a headline - a phrase. This sentence-based approach allows to make assumptions about the quality of the tracked information, as it would be improper to assume a text is being (thoroughly) read, when the gaze point is pointing on a text. It could be, for example, just skimmed, as stated by Kollmorgen and Holmqvist [29]. We further describe this problem in Section 3.5.

Figure 3.7 (a) illustrates the tracked metrics: Time, Sentence, Component Position, Level of Detail, Section/Component Name and Image.

3.3.2 Adaptive Text Visualizations

A$^+$CHIS also offers multiple document visualization approaches, one of them being the *WordCloud*. It contains keywords, that are pre-generated via topic modeling of the corresponding document chapter, as described by Shao et al. [30]. Similarly to the adaptive information graphics, they also offer 3 aggregation forms, which are shown in Figure 3.5.

The tracked metrics include Time, Word (being the smallest piece of information which is saved), Component Position, Level of Detail, Section/Component Name and Image, as shown in Figure 3.7 (b).



(a) Aggregation: Low (b) Aggregation: Mid (c) Aggregation: High

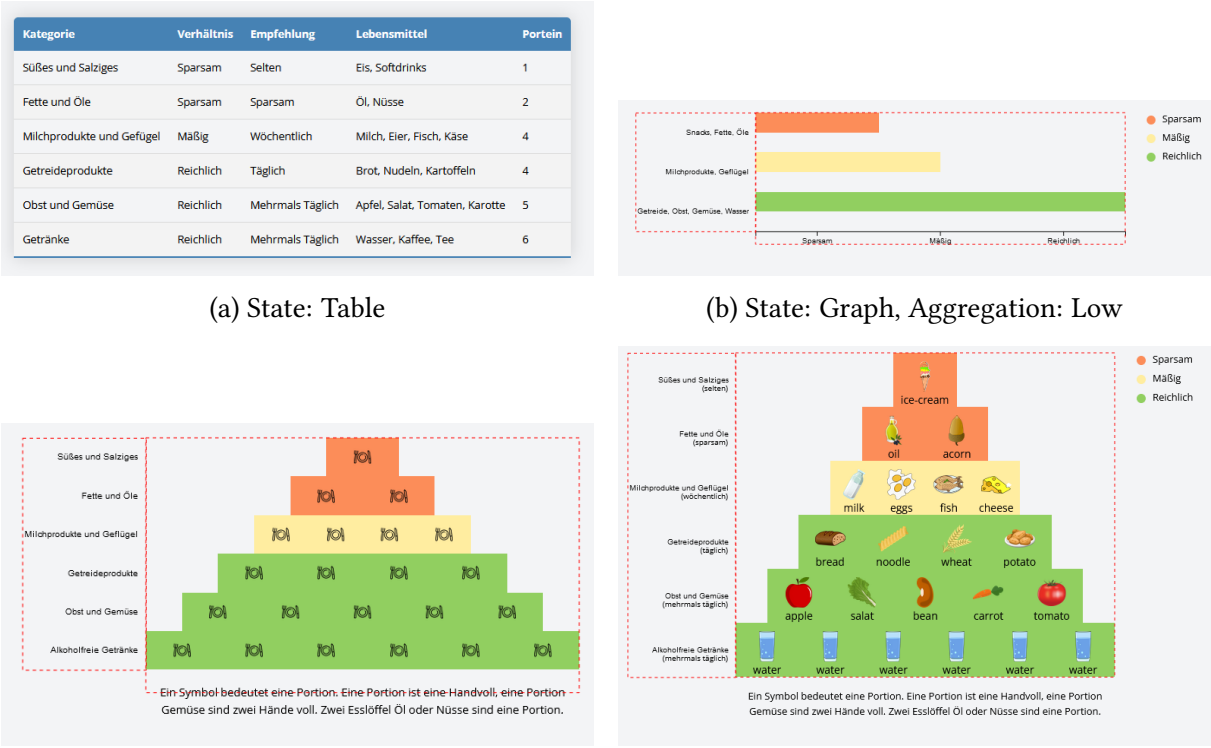
Figure 3.5: All aggregation forms of the *WordCloud*-visualization.

3.3.3 Adaptive Information Graphics

A$^+$CHIS currently offers a variety of information graphics with multiple aggregation forms. The intent of these aggregation forms is to display graphs (and document visualizations) with different Levels of Detail (LoD) to match the visualization literacy of the current user. Furthermore, these visualizations offer 2 states: (1) textual representation in form of a table and (2) graphical representation, offering another dimension for adaptation. For this work, the *Nutri Pyramid* graph was selected. Depending on user selection, it displays a nutrition pyramid in different graphical and textual representations, defining our state and aggregation form: A table (Figure 3.6 (a)), a bar chart (Figure 3.6 (b)) and a pyramid (Figure 3.6 (c) and (d)). In context of our dimensionality model, the tracked metrics for this visualization include Time, Graph Area, Component Position, LoD (=aggregation form), Section/Component Name and Image, as shown in Figure 3.7 (c).

The smallest piece of information for this visualization type is defined by the Graph Area. Figures 3.6 (b), (c), and (d) show red rectangles around the graph itself and the axis descriptions,

which define the implemented areas. Depending on where the gaze point is located inside the component, the corresponding area is saved as a short string.



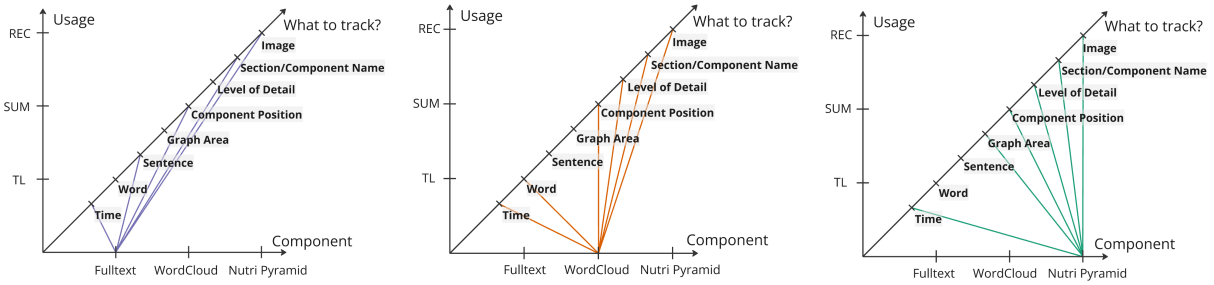
(a) State: Table

(b) State: Graph, Aggregation: Low

(c) State: Graph, Aggregation: Mid

(d) State: Graph, Aggregation: High

Figure 3.6: All states and aggregation forms of the *Nutri Pyramid*-visualization.



(a) Fulltext

(b) WordCloud

(c) Nutri Pyramid

Figure 3.7: The 3 main components and tracked metrics in our dimensionality model.

3.4 Usage

In this section, we introduce our main data structure as a basis for 3 applications: (1) Timeline Analysis, (2) Summative Analysis and (3) Recommendations. While (1) and (2) are user/expert

Table 3.2: A detailed description of the *FsElement*-Object.

Instance	Type	Source/Computation	Comment
section	string	tag, level 3	
modal	boolean	tag, level 2	
component	string	tag, level 1	
text	string[]	tag, level 0	[] used for summarization
component_type	string	tag	e.g. div, svg, button
text_category	string	tag	text, word, sentence, graph
image	string[]	backend/component matching	[] used for summarization
from	Date	backend/fixation computation	
to	Date	backend/fixation computation	
duration	integer	to - from	ms
time	string	frontend/component tracking	
fixation	boolean	initial value: true	Used for timeline visualization.
focus	string[]	if component is one of the main components: [FOCUS, COMPONENT_ACC] else [NAVIGATION, NAV_ACC]	
state	string	frontend/component tracking	
aggregation	string	frontend/component tracking	
id	integer	backend	

level visualizations, (3) acts as a system implementation, which is intended to generate meaningful data for later usage.

Data Structure

Our data structure is an array of *FsElement*-objects, including tracked metrics and inferred values from the tracked data. Table 3.2 offers a detailed list of the instances of the fixation object and how they are obtained. One *FsElement* (Fixation-Saccade-Element, as it can represent a fixation or a saccade) is the combination of a maximum of 4 DOM-tags from Table 3.1. This array is a key implementation milestone and the basis for all usage scenarios.

3.4.1 Timeline Analysis and Summarization

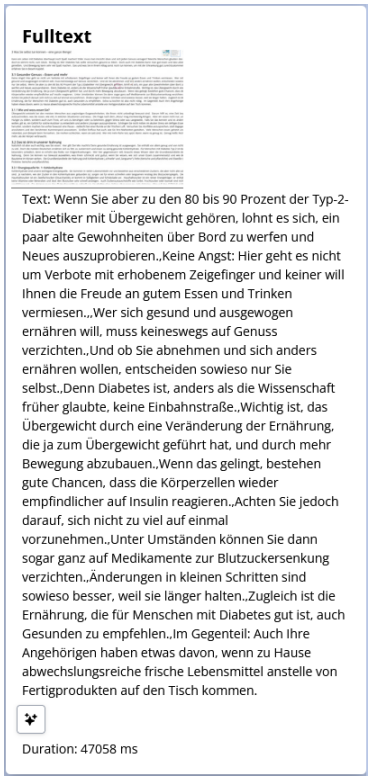
As explained in Section 2.4, explainabilty is an important instrument for systems to offer the user information about the computation model, regardless of whether it is used for recommendations, adaptations or summaries. As pointed out by Blaschek et al. [31], due to the temporal nature of gaze data, a timeline visualization is appropriate. In Figure 3.12 (a), the utilized tracked values are shown in our dimensionality model. Additionally, we compute the duration and focus state (NAVIGATION/FOCUS) of each fixation. Beside a time-dependent interactive lineplot for the gazepoints in x- and y-direction, this work introduces a card-based fixation visualization (Figure 3.8), where viewed components and information about them (as listed in Table 3.2) can be retraced.

Figure 3.8: Our timeline visualization with fixations in the *Fulltext* component. The lineplots at the bottom show the gaze movements in the x- and y-direction respectively, while the cards show individual fixations.

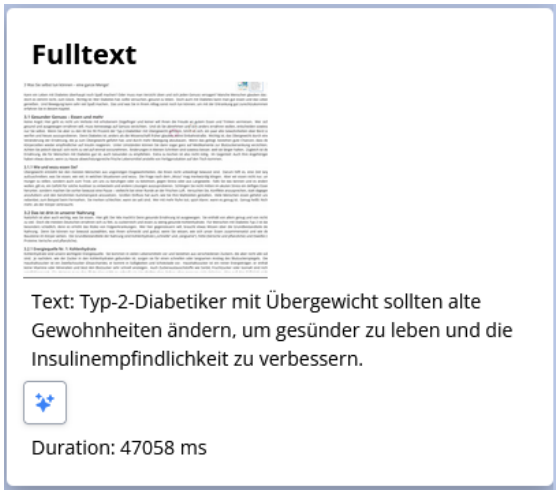
Although the display of all fixations is a starting point for investigating the exploration history, it lacks overview. Because of that, a grouping function has been implemented. For textual representations (in our case the *Fulltext*-component), it takes all fixations from an AOI (for us, a component in focus), and creates a set of read sentences and phrases. Graphical representations and document visualizations, as the *Nutri-Pyramid* and *WordCloud*, are grouped by aggregation form. This grouping function does not only offer a better understanding of the timeline, as shown in Figure 3.9, but also allows to compute further metrics, as mentioned in Section 3.5.

A set of tracked texts for text-based grouping does not necessarily follow the linear structure of the read text, as typical reading behavior also incorporates regressions (see Figure 3.2). It is

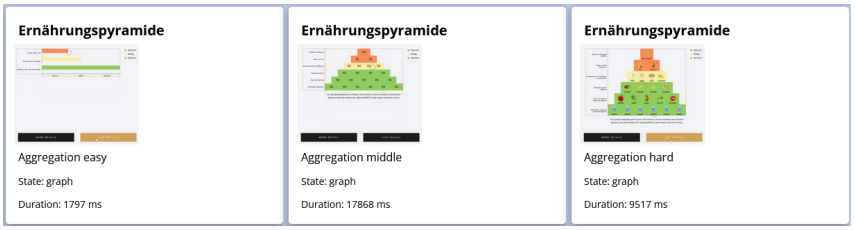
also possible, that the user does not comprehend the read text passage during exploration. For this matter, the option of summarization by an LLM has been implemented, which is taking the set of sentences, together with a pre-defined message as a prompt and returns the summary as an answer. A detailed description of the LLM-usage is given in Section 4.1.4. The navigation grouping returns screenshots of the 2 longest-viewed components and points to the section, the user has navigated to.



(a) Text grouping without LLM-summary.



(b) Text grouping with LLM-summary.



(c) Graph grouping.

Figure 3.9: Groupings of text (a,b) and graph (c) components, including LLM-summary.

3.4.2 Summative Analysis

In this Section, we introduce the *Hierarchic Knowledge Collection* graph, which combines knowledge retrieval and the display of viewing metrics from the captured ET-session. To handle the amount of data, which is generated - even with short ET-sessions - and to offer meaningful insights, two hierarchies are introduced, as shown in Figure 3.10.

In Figure 3.12 it can be observed, that almost all of the tracked metrics are used for this visualization, as well as most of the instances of our fixation array.

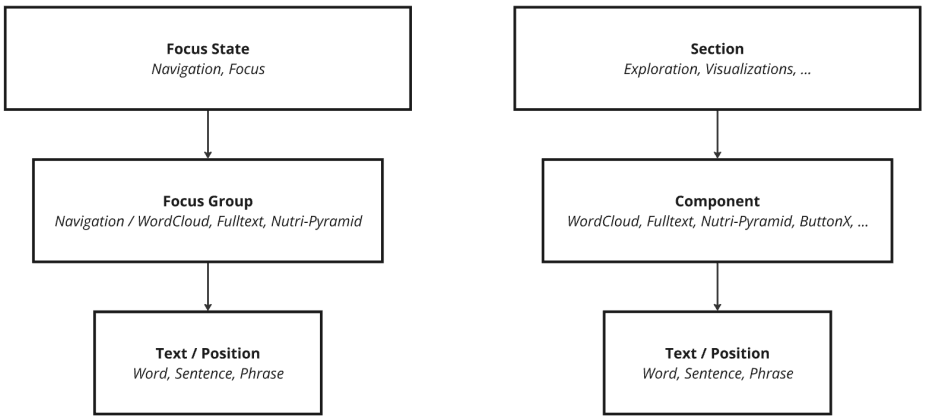


Figure 3.10: The hierarchies for the Knowledge Collection chart.

Even though a Treemap would be an option to display some of this data in a clear way, the data is shown as a hierarchic bar-chart. This has multiple advantages for us: (1) concerning accessibility, value and extremum retrieval are better possible. (2) they are among the most used visualization methods, based on news outlets and school curricula, as identified by Lee et al. [32]. (3) the layouting does not always allow a consistent presentation of additional information, such as texts or lines, inside the Treemap. Figure 3.11 shows our knowledge graph in its initial state.

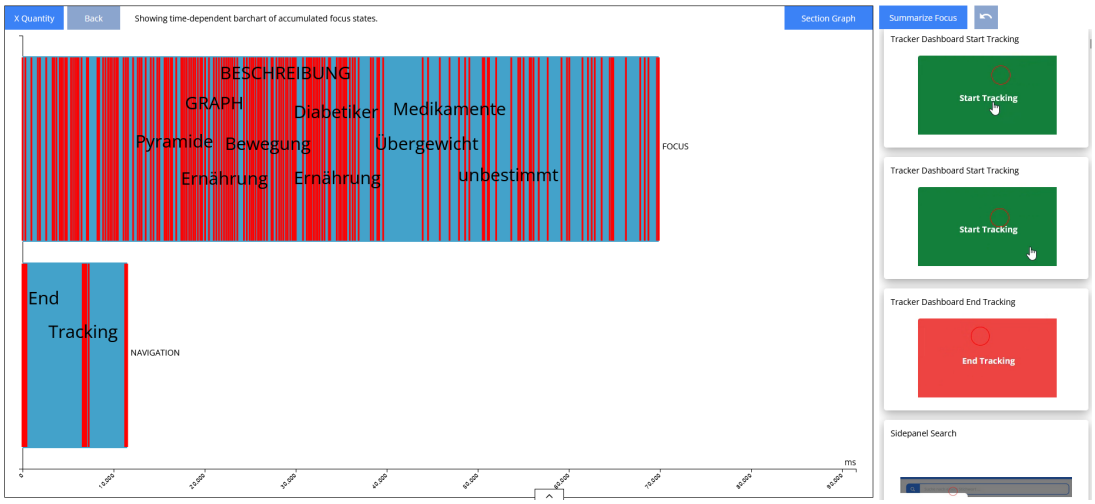


Figure 3.11: The initial state of the knowledge graph at hierarchy level 1.

This chart offers various ways to visually analyze and retrieve knowledge about the user's exploration journey. The scrollable fixations list, which can be found at the right-hand side, shows all fixations in the current hierarchy level. With detailed information on click, it lets the user revisit important components, words or sentences. The list can be summarized, similarly to the timeline visualization, to achieve a better overview. The x-axis can be switched between accumulated quantity (number of fixations) and time (duration). Both are measures for user interest. The graph in Figure 3.11, e.g., shows the time spent during navigation and looking at various components. The next hierarchy level can be reached by clicking a bar.

Individual fixations are shown by vertical lines inside the bars. This is particularly interesting on the third/last hierarchy level, as the user (or evaluator) is able to identify revisits of a certain graph section or text, as shown in Figure 5.8. By clicking a line, a popup opens with the position of the fixation inside the timeline visualization, linking these tools together and offering a temporal categorization.

The user is provided with a WordCloud inside the bars, generated by an LLM, as a preview for the underlying hierarchy children. Every entity in the second level (for example extracted texts during time spent in the *Fulltext* component) is – together with a pre-defined message – sent to the LLM, which returns the 5 most important words, as described in Section 4.1. This offers a second level of summarization.

3.4.3 Component- and Content-based Recommendations

The recommendation system is the key element between tracked content and the adaptation to the user. For our recommendations, we make use of all content- and component-based records in our dimensionality model, as visualized in Figure 3.12.

To achieve the goal of adaptation, the A$^+$CHIS system is based on components, some of which have multiple aggregation forms or represent the same information graphically/textually. To describe these components, a 4-dimensional Feature Vector (FV) with the elements <*text, graph, aggregation, state*> is created with pre-defined values for every component. *Text* and *graph* describe the textual and graphical tendencies of the component. *Aggregation* represents the LoD and *state* indicates, whether the visualization is textual (e.g., a table) or graphical. In this thesis, values (between 0 and 1) are selected to the best of our knowledge. In a production environment, domain experts would have to rate each component individually. Table 3.3 displays a selection of rated components from A$^+$CHIS, which have been used to evaluate this part of the implementation.

To compute the user u's FV $\langle t,g,a,s\rangle_u$, the fixation array is first cleaned up from missing values and then matched with the existing components list, based on *aggregation* and *state* values, to retrieve the *text* and *graph* dimension. After this, the weighted average of all FVs is computed by using the fixation times $\{w_i\}_{i\in I}$ as weights (in milliseconds) and I as the index set of fixations:

$$\langle t,g,a,s\rangle_u = \frac{\sum_{i\in I}\langle t,g,a,s\rangle_i w_i}{\sum_{i\in I} w_i}. \tag{3.1}$$

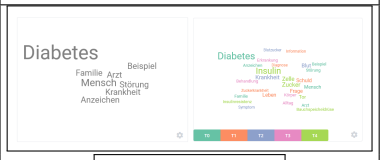
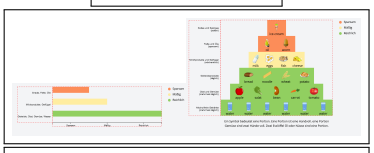
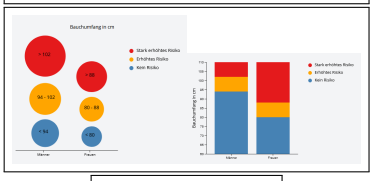
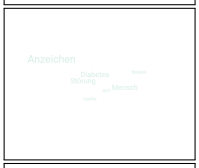
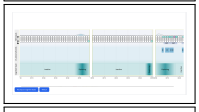
Afterwards, the nearest neighbors between component list and the user's FV are returned.

To offer targeted recommendations regarding the content of a document, Tf-idf (term frequency - inverse document frequency), which was conceptualized by Jones [33], combined with cosine similarity is implemented. Tf-idf is the product of the term frequency (Equation 3.2, occurrences of a term t in a document D, divided by the max number of terms in the document) and the inverse document frequency (Equation 3.3, logarithm of the number of documents N in the corpus devided by number of documents containing t):

$$\mathrm{tf}(t,D) = \frac{\#(t,D)}{\max_{t'\in D}\#(t',D)} \tag{3.2}$$

$$\mathrm{idf}(t,D) = \log\frac{N}{|\{d\in D|t\in d\}|} \tag{3.3}$$

Table 3.3: A selection of implemented components with the corresponding FV and images.

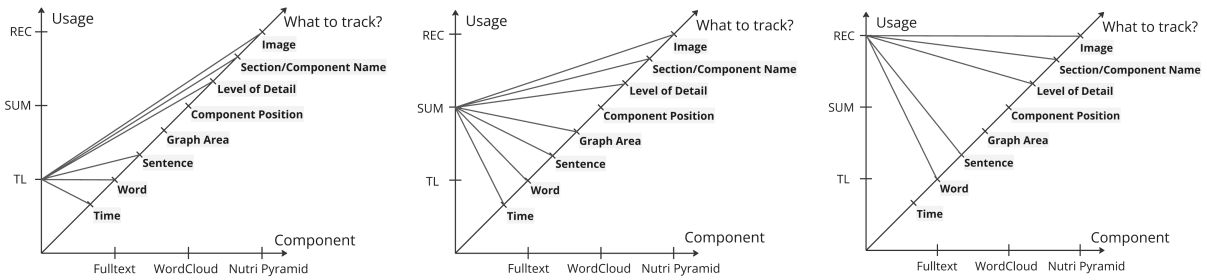
Component Name	Screenshot	Vector			
		text	graph	aggregation	state
WordCloud		0.5	0.5	0-1	0/1
TopicCloud		0.6	0.5	1.0	1.0
Food Pyramid		0.1-0.8	0.2-0.9	0-1	0/1
Belly-Viz		0.0-0.8	0.2-1.0	0-1	0/1
Fulltext		1.0	0.0	0.0	0.0
Split View		0.9	0.1	0.0	0.0
Theme Pyramid		0.6	0.4	0.5	1.0
HistoryCloud		0.5	0.5	0.5	1.0
Dashboard		0.2	0.8	1.0	1.0
Tilebar		0.3	0.7	1.0	1.0
Word Frequency		0.3	0.7	0.0	1.0

$$\text{tf-idf}(t,D) = \text{tf}(t,D) \quad \text{idf}(t,D)) \tag{3.4}$$

After vectorization, which includes computing Tf-idf for all words in the document (provided by A$^+$CHIS) relative to our query (the extracted texts), the cosine similarity

$$\text{cosine similarity}(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} \tag{3.5}$$

between our query vector A and the document vectors B is computed, by measuring the angle between them. This method returns ranked similarities between our query of extracted texts and chapters in the document, which could help making suggestions regarding interest, or point to not-yet discovered regions of the document.



(a) Timeline Analysis (b) Summative Analysis (c) Recommendations

Figure 3.12: The 3 main usage examples and tracked metrics in our dimensionality model.

3.5 Further Metrics

With our tracking integration as a basis, further metrics, which are beneficial for recommendations and pattern analysis, can be implemented comfortably, as described in the following sections.

3.5.1 Text/Exploration Coverage

Bieder et al. [34] state, that the most robust way to detect reading is to use per-word scanning. However, a per-word approach comes with drawbacks, such as inaccuracies based on tracking noise. As we extract texts and save them with corresponding fixation times, we can analyze the duration spent on a sentence/word and compare them to set data structures, for example topic lists (*WordCloud*) or chapters (*Fulltext*). The results are represented by a percentage value for text coverage and visualized by a color-coded document representation, similar to *Seesoft* by Eick et al. [35]. This method is also applicable for whole components or graphs.

3.5.2 Area Switches

We define area switches as transition between sections in a graph, as displayed in Figure 3.6, by dashed red lines, similarly to Harsh et al. [10]. While they did not find statistically significant differences between the users' expertise level solely on graph features, differences arose when comparing viewing times of graph and contextual elements, such as descriptions or legends. With a proper definition of AOIs, this could help distinguishing between experts and beginners, which would subsequently ease the process of recommending correct graph types, in terms of difficulty.

3.5.3 Aggregation Switches

Although not directly connected to ET – because they are done by clicking buttons – aggregation switches could also offer an indication of expertise. Depending on the pre-set aggregation of graphs (e.g., high), we are able to analyze for directional patterns, such as always selecting the easiest display form, or vice versa.

CHAPTER 4

Implementation

This chapter is showcasing implementation details based on the concepts presented in Chapter 3. First, we present the hard- and software framework. In the second subchapter, we explain necessary steps for the data collection, such as implementing the ET-device and modifications of the frontend. After that, the fixation calculation is explained. Lastly, we show how we implement the visualizations and recommendations using the calculated fixation array.

4.1 Hard- and Software Framework

This section presents the hardware, libraries, frameworks and tools which are used for implementation.

4.1.1 Eye Tracker

The eye tracking device used in this work is the *Tobii Pro Spark*[1] with the driver version 2.2.3.0. Table 4.1 lists the (for us) most important technical specifications of the eye tracker, taken from the product description[2].

The installation of the tracker is straightforward: First, *Tobii Pro Eye Tracker Manager*[3] has to be downloaded, which is necessary for setup and calibration. Second, the *tobii-research* library needs to be added for Python bindings. After these steps, we can continue with implementation and tracking.

[1] https://www.tobii.com/products/eye-trackers/screen-based/tobii-pro-spark
[2] https://go.tobii.com/tobii-pro-spark-product-description
[3] https://connect.tobii.com/s/etm-downloads

Table 4.1: Technical specifications of the Tobii Pro Spark Eye Tracker.

Sampling frequency	60 Hz (default) or 33 Hz
Precision	0.26° RMS in optimal conditions (using built-in filtering)
Accuracy	0.45° in optimal conditions
Binocular eye tracking	Yes
Data sample output	Timestamp, Gaze origin, Gaze point, Pupil diameter
Operating distance (mounted on Screen)	45 cm to 95 cm from the eye tracker
Freedom of head movement (width x height)	At 50 cm distance: 20 cm × 20 cm
Optimal screen size	Up to 27” (16:9 aspect ratio)

4.1.2 Frontend Framework

The frontend framework used by the A$^+$CHIS system is the TypeScript, HTML and CSS based Angular [36] version 13.3. While numerous libraries play together to help creating and visualizing the frontend in A$^+$CHIS, the usage of *RxJS* [37] should be noted, as it is used for Observable and Subscription patterns for callback-based events. We also use *Tailwindcss* [38] for design and *D3* [39] for graph creation.

4.1.3 Backend Framework

The backend framework, the Python-based Django / Django REST Framwork [40], is especially relevant for us, as our ET-device supports this language. Similarly to the frontend, we also use multiple libraries for our implementation. Notable libraries are *django-channels*, which is used to establish a WebSocket connection, *FFMpeg* [41] for extracting component screenshots and *scikit-learn* [42] for text-based similarity measures.

4.1.4 Inclusion of Large Language Models

The rising importance of LLMs in peoples’ everyday lifes motivates to find meaningful ways to incorporate these models into applications. ChatGPT from OpenAI [43] is currently one of the most popular chatbot, which was thus selected for usage.

We implement 2 different use-cases, which are further described in Chapter 3, both using the 4o-mini model: (1) text summarization and (2) extracting 5 relevant tems from the given text. The following system and user prompts mainly address the *Error Identification* and *Prompt Improvement* categories of prompt patterns, identified by White et al. [44]. According to the OpenAI documentation[4], the `system` prompt is prioritized and used to give instructions to the model, while the `user` prompt, which is in German to cohere with the language used in

[4]https://platform.openai.com/docs/guides/text-generation

A$^+$CHIS, is used to request an output from the model.

(1) Text Summarization

```
Role: system
You will summarize and simplify the given text in German. Make a super
short comment no exceeding 150 characters. Use simple language. Do not
mention being an AI or a chat assistant. Ignore redundant commas. Keep
the summary concise, ideally in one sentence.

Role: user
Fasse den folgenden Text zusammen und vereinfache ihn: {text}
```

(2) 5 Relevant Words

```
Role: system
You will return a list of the 5 most relevant words of the given text
in German. Use the nominal form. Ideally use words that are included in
the text. Do not mention being an AI or a chat assistant. Do without
enumeration and only use commas without whitespace as separator.

Role: user
Gib mir die 5 relevantesten Wörter aus diesem Text: {text}
```

4.2 Data Collection

The first step, after the installation of the ET-device, is to gather gaze data, extract information from DOM-elements and structure this information, so that it can be employed for our recommendations and visualizations.

4.2.1 General Data Flow

The data pipeline is sketched and explained in Figure 4.1. The raw gaze data from the eye tracker is first checked for missing values before calculating the center of the left and right eye coordinates, as explained in Section 4.2.2. The screen recording and component tracking happens in the frontend during then ET-session. After a session is finished and when requested, the tracked components and the fixations, which are calculated in the frontend, are matched and sent back to the frontend for further filtering and calculations.

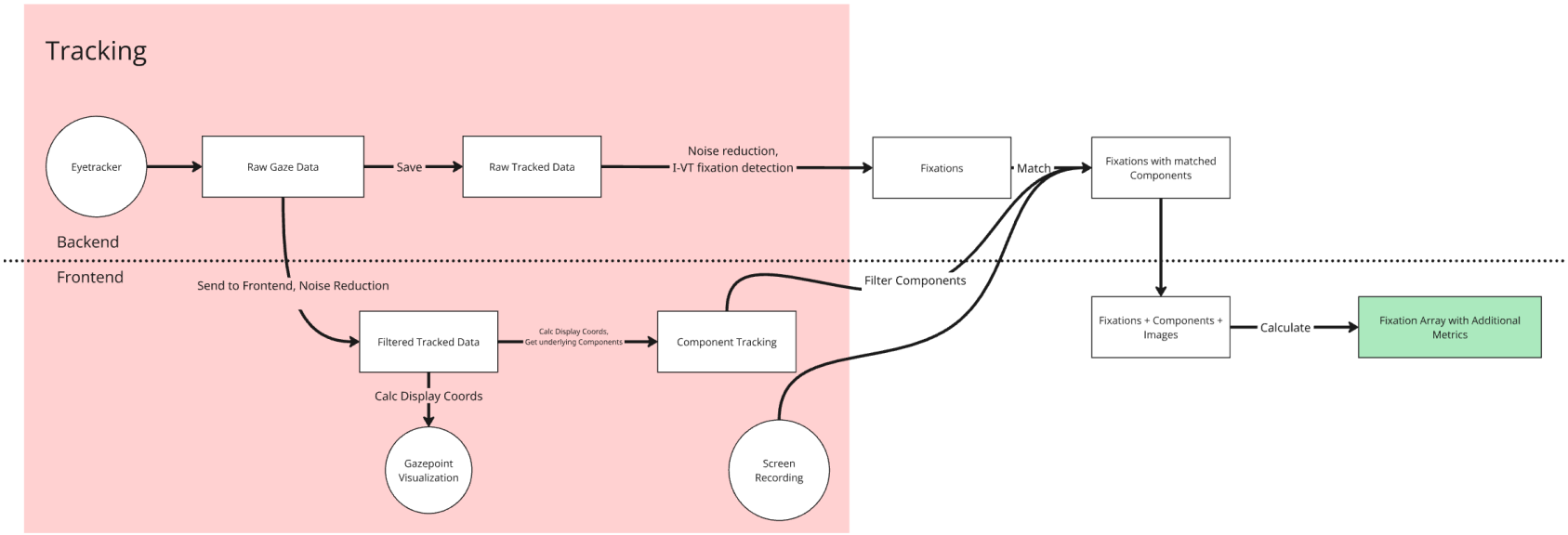


Figure 4.1: A sketch of the ET-data pipeline. The raw gaze data is stored in the backend and sent to the frontend. In the frontend, noise reduction is performed for a smoother gaze point visualization and the data is also used for identifying DOM-components, which are, together with a screen recording, sent back to the backend at the end of the ET-session. Upon request, fixations are calculated after noise reduction and matched against the tracked components in the frontend, including extraction of the video frames for the component visualization.

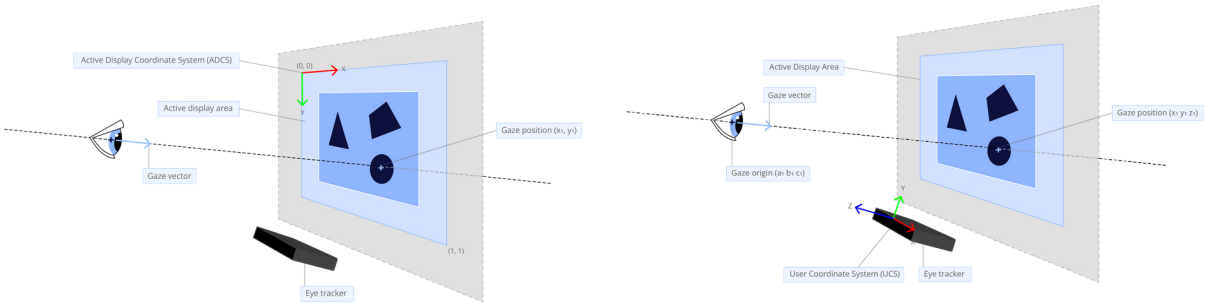
4.2.2 Introducing ET to the A$^+$CHIS Project

As the *Tobii Pro Spark* does not offer language support for JavaScript, but does support Python, the tracker is connected to the backend. It provides its data via a callback function. After creating a new Django app, we need to install *django-channels*, which is the native library to implement a WebSocket connection. After establishing a connection, we implement 3 modes: (i) `gp` for continuous retrieval of the gaze point, (ii) `usercalibration/displayarea` for calling the *Eye Tracker Manager* directly from the frontend, and (iii) `unsubscribe` for ending and saving the session, as shown in Listing 4.1.

```
1  def receive(self, text_data):
2      global et
3      global points_temp_storage
4
5      if(et==None):
6          eye_trackers = tr.find_all_eyetrackers()
7          et = eye_trackers[0]
8
9      text_data_json = json.loads(text_data)
10     mode = text_data_json['mode']
11
12     if mode == 'gp':
13         points_temp_storage = []
14         et.subscribe_to(tr.EYETRACKER_GAZE_DATA, gaze_data_callback, as_dictionary=True)
15         return
16
17     if (mode == 'usercalibration') or (mode == 'displayarea'):
18         call_eyetracker_manager(mode)
19         return
20
21     if mode == 'unsubscribe' and et != None:
22         et.unsubscribe_from(tr.EYETRACKER_GAZE_DATA, gaze_data_callback)
23         UCS_serializer = serializers.PointUCSSerializer(data=
24           points_temp_storage,
25           many = True,
26         )
27
28         if UCS_serializer.is_valid():
29             UCS_serializer.save()
```

Listing 4.1: The receive function of the Websocket.

The `gaze_data_callback` function (Listing 4.1, line 22) retrieves 3 pairs of coordinates from the tracker: L/R (left and right) gaze point on the display area, L/R gaze point in the user coordinate system and L/R gaze origin in the user coordinate system. For all of them, the center between left and right eye is calculated and gap values are filled with the previously valid coordinate. The Active Display Coordinate System (ADCS) is necessary to display gaze points in the frontend and match with DOM components, while the User Coordinate System (UCS) is needed to calculate the velocity between gaze points. The difference between these coordinate systems is shown in Figure 4.2. All resulting coordinates are saved in the backend with time and session ID and the ADCS coordinates are sent to the frontend.



(a) Active Display Coordinate System (b) User Coordinate System

Figure 4.2: The coordinate systems used by our ET-device, taken from the *tobii* documentation[5].

4.2.3 Tracking of DOM Elements

Having the current gaze coordinates in the frontend, the next step is to label each relevant DOM-component. We do this by introducing a hierarchy with 4 levels, as explained in Section 3.3. HTML-elements that should be tracked are manually tagged with the section name (which is the general area of the app the user is currently navigating in), level, name, type and internal ID of the element, as shown in Figure 4.3.

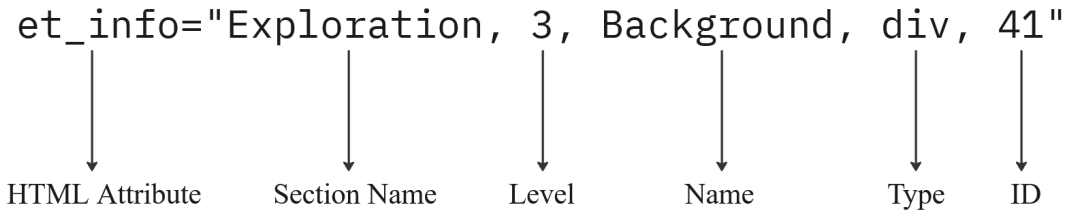


Figure 4.3: The structure of a tag for component identification.

To track these tagged elements (and to visualize the gaze point), we first need to translate the display coordinates, which are initially provided as a relative screen position by the ET-device. In Listing 4.2 we first filter the raw data with a low-pass filter to reduce noise, using $\alpha = \frac{T}{\mathrm{h}}[45]$ which is the quotient of the time constant T and the sampling interval h of the eye tracker (1/60 s). α is then multiplied with the previously filtered point $\hat{\mathrm{p}}_{n-1}$, added to the current point p_n and divided by $1 + \alpha$:

[5]https://developer.tobiipro.com/commonconcepts/coordinatesystems.html

$$\hat{p}_n = \frac{p_n + \alpha \hat{p}_{n-1}}{1 + \alpha} \quad [45] \quad (4.1)$$

As explained by Olsson [45], if we set time constant too high, the gaze movement would be slowed down on saccadic movements. A too low T would make the filtering redundant. We found, that T = 0.2 offers the best results for visualization purposes.

After that, the x and y coordinates for the gaze point are computed by multiplying x with the total width of the screen (as the browser is borderless on the sides) and y with the height, corrected by the individual height of the topbar. The coordinates for displaying the gaze point are additionally corrected to fit the size of the representing circle, as shown in Listing 4.2.

```
this.webSocket.onmessage = (e: any) => {
  const data = JSON.parse(e.data);

  var filteredData: Point = this.filterGP({x: data.x, y: data.y});

  var gp_f: GazePoint = {left_x: ((filteredData.x * SCREEN_TOTALWIDTH - GP_OFFSET) + "px"),
                         top_y: (((filteredData.y * SCREEN_TOTALHEIGHT) - WINDOW_TOPBAR_HEIGHT - GP_OFFSET) + "px"
                                 ),
                         real_x: (filteredData.x * SCREEN_TOTALWIDTH),
                         real_y: ((filteredData.y * SCREEN_TOTALHEIGHT)-WINDOW_TOPBAR_HEIGHT),
                         time: data.time}

  this.gazePositionSubject.next(gp_f);
}
```

Listing 4.2: The calculations required for correct display of the gaze point.

The function which listens to the `gazePositionSubject` displays the gaze point and returns the underlying objects by using `elementsFromPoint()`[6], which is a method of the `Document` interface of a web page. By filtering for our tagged elements, we receive an array of elements in a set interval and can extract texts and measurements based on component type.

4.3 Fixation Calculations

These calculations happen in the backend upon request after a finished ET-session. After calculating via the I-VT-algorithm, as described in Chapter 3, the fixations are matched with corresponding DOM-elements from the frontend.

4.3.1 Fixations

First, we import the saved raw gaze data containing all gaze positions from both coordinate systems, including timestamps, in a `pandas.DataFrame` and run a low pass filter with $T = 0.01$ over the data. After this, the fixations are calculated based on the pseudocode in Algorithm 1 taken from Salvucci and Goldberg [26].

[6] https://developer.mozilla.org/en-US/docs/Web/API/Document/elementsFromPoint

Algorithm 1 Computing fixations based on the I-VT algorithm from Salvucci and Goldberg [26].

```
function I-VT(GazePoints, Velocity Threshold)
    for all Point in GazePoints do
        calculate point-to-point velocities
    end for
    for all Point in GazePoints do
        if Velocity < Velocity Threshold then
            label as fixation point
        else
            label as saccade point
        end if
    end for
    Collapse consecutive fixation points into fixation groups, removing saccade points
    Map each fixation group to a fixation at the centroid of its points
    return fixations
end function
```

These calculations happen point-wise in the `DataFrame`. By constructing a triangle between the coordinates of the user's eye, the current point and the previous point, the point-to-point velocities can be calculated by using the timestamps from the gaze data provided by the tracker, the distance between the two gazepoints and the angle between the gaze trajectories. After labeling, the centroids of the fixation groups, which are separated by saccades, are determined.

4.3.2 Matching with DOM Elements

By using `elementsFromPoint()` in the frontend, all DOM elements underneath that point get extracted. Although the frontend only delivers those elements which were previously tagged, we still need to sort them according to our pre-defined levels, as there might be a modal with another component underneath it. The pseudocode in Algorithm 2 shows this filtering algorithm, which returns a vector following our desired pattern: 0 (optional) $\rightarrow$ 1 (optional) $\rightarrow$ 2 (optional) $\rightarrow$ 3 (always given). The more of the optional levels 0 and 1 are included, the more precise is the extraction and tracking.

After filtering, we loop through the array of fixations and match them with the element group which is the nearest to the fixation, according to the recorded timestamp. This timestamp is also used to extract a screenshot from the screenrecording and crop it based on the extracted `clientRect` with *ffmpeg-python*, which is a library offering Python bindings for *FFMpeg* from Tomar [41]. The result is a finished array of fixations including extracted information, which we can send now to the frontend for application.

Algorithm 2 Filtering the DOM-elements levels.

```
function FILTER(Element Groups)
    Filtered Element Groups = []
    for all Element in Element Groups do
        for all DOM Element in Element do
            if Level > Previous Level then
                add to Filtered Element Groups
            else
                dismiss
            end if
        end for
    end for
    return Filtered Element Groups
end function
```

4.4 Applications

This section shows implementation details for the timeline visualization, information cloud and recommendations, based on the concepts presented in Chapter 3. Additional values such as *focus*, *fixation* and *duration* (Table 3.2) are computed upon receiving the fixations array from the backend.

4.4.1 Timeline Analysis

The timeline (as introduced in Section 3.4.1) is a horizontally scrollable interface based on HTML/CSS with 3 main features: Fixation Grouping, Interactive Gaze Graph and LLM-based Summarization. While the width of the fixations is static for better clarity, the saccade length is based on the time between two fixations.

Fixation Grouping

The fixation grouping function takes the fixations which are labeled as NAVIGATION or FOCUS (see Table 3.2 with the corresponding component and accumulator and joins them together. Based on the component type, the text or aggregation form is grouped. Algorithm 3 shows a pseudocode for the FOCUS grouping function, which presents the basic functionality. In the actual code, more error catching logic was necessary to ensure meaningful groupings. The NAVIGATION grouping works similarly, without taking different component types into account but grouping the whole navigation process.

Interactive Gaze Graph

The interactive gaze graph consists of 2 separate time-dependent graphs (x and y coordinate

Algorithm 3 The grouping of FOCUS-tagged fixations.

```
function GroupFocus(Fixations)
    GroupedFixations = []
    TempArray = []
    for all Fixation in Fixations do
        if Fixation.Focus is FOCUS then
            append Fixation to TempArray
        end if
        if Fixation.Focus is NAVIGATION then
            append Fixation to GroupedFixations
            if TempArray not empty then
                if Fixation.Component is Graph or WordCloud then
                    group TempArray to 1-3 fixations based on aggregation type
                end if
                if Fixation.Component is Fulltext then
                    group TempArray to 1 fixation based on text
                end if
                append to GroupedFixations
            end if
            TempArray = []
        end if
    end for
    return GroupedFixations
end function
```

of the screen) including all gaze positions from the tracking session. These graphs are implemented with D3 and offer interactivity on hover, by showing a popup and scrolling the fixation timeline to the hovered fixation. The scrolling in angular has been implemented by extracting the gaze point time from the mouseover event in the graph and a `scrollTo()` function injected in every fixation card template. This function checks if the time is in between the fixation's timeframe and uses `scrollIntoView()`[7] to display the right element.

LLM-based Summarization

The LLM-based summarization happens at the same time as the group of fixations by sending the summarized text to the backend asynchronously, adding the text to a prompt as described in Section 4.1 and showing it upon request from the user. This makes sure that not every summarization requires a request to ChatGPT.

4.4.2 Summative Analysis

The concept behind the multi-layer bar charts in the *Hierarchic Knowledge Collection* graph (as introduced in Section 3.4.2 is the hierarchic representation of our fixation array in form of a tree. One element of this tree is shown in Listing 4.3. Level indicates the hierarchy tier, count is the x-axis value, dimension the y-axis value, words is used for the Word Cloud inside the graph and times is used for the horizontal fixation lines. Parent and children are used to navigate through the data structure.

```
interface vis_array {
  level: number,
  parent: string,
  count: number,
  dimension: string,
  children: vis_array[],
  words: string[],
  times: any[]
}
```

Listing 4.3: One element of the visualization hierarchy, used for the Hierarchic Knowledge Collection graph in the Summative Analysis.

The graph is then created using *D3* [39] with the user-selected hierarchy (focus/section), counter (# of fixations or time) and level. Interactivity is added by sorting, hover/click events and smooth transitions. The right-hand side of the chart lists all fixations included in the selected hierarchy level and uses the same summarization function as the timeline visualization.

Callback functions are implemented to provide additional information to the user and enable the exploration of the hierarchy. Clicking a fixation from the list shows all saved details of that fixation in a modal, clicking a horizontal red line in the barchart shows the position of

[7] https://developer.mozilla.org/en-US/docs/Web/API/Element/scrollIntoView

the fixation inside the timeline component in a modal and clicking on a bar brings the user to a deeper hierarchy level while also filtering the list of fixations.

The WordCloud inside the bars shows the most important terms from the deeper hierarchy sections and is implemented by passing all texts to the LLM in the backend (as described in Section 4.1) with an asynchronous function upon graph creation and setting the layouting borders of the cloud generation algorithm to the respective borders of the bars.

4.4.3 Recommendations

The computation of the recommendations happens upon user request in the frontend. The array of fixations with all additionally computed values is sent to the backend and and irrelevant (i.e., NAVIGATION-tagged) components are filtered out. The results of the content-based recommendations including the own FV, as well as component-based recommendations are sent back to the frontend and visualized with the intent of proving the functionality of the concepts and providing the values for future adaptations.

Content-Based

For content-based recommendations, we combine Tf-idf with cosine similarity, as explained in Chapter 3. We make use of the already existing class `TfidfVectorizer` and function `cosine_similarity` from the *sklearn* library of Pedregosa et al. [42]. The vectorizer returns a matrix with all weights of our search query, which are the extracted texts from the tracking, and those of the whole input document, which is a collection of documents (chapters). With the cosine similarity, we compute the similarities between the document (chapter) vectors and the vector of our query. This process is shown in Listing 4.4, with the request (containing the document-id) and query as an input, and the similarities, which is a sorted list of the cosine similarity function as an output.

```
1  def compute_similarity(request, query):
2      did = request["did"]
3      document = apchisexplore.models.Document.objects.get(id=did)
4
5      texts = document.text_set.all()
6      texts_list = []
7      for text in texts:
8          texts_list.append(' '.join([stemmer.stem(word) for word in word_tokenize(text.content_sentence_based.lower
               ()) if word not in german_stopwords]))
9
10     texts_list = [query] + texts_list
11
12     vectorizer = TfidfVectorizer(stop_words=german_stopwords)
13     tfidf_matrix = vectorizer.fit_transform(texts_list)
14     cosine_similarities = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:])
15
16     similarities = {texts[i].id: cosine_similarities[0][i] for i in range(len(texts))}
17
18     return similarities
```

Listing 4.4: The computation of content-based similarities based on Tf-idf and cosine similarity.

Component-Based

To compute the component-based recommendations, we first map the tracked components against a static list of implemented components (Listing 4.5) to retrieve the pre-defined *graph* and *text* values. After this, the user's own FV is computed by using the weighted average with the viewing times as weighs, as described in Equation 3.1. In the end, the 3 nearest neighbors based on the L_2 distance are computed.

```
componentlist = [
    {"name": " WordCloud", "text" : 0.5, "graph" : 0.5, "aggregation": 0.0, "state": 1.0},
    {"name": " WordCloud", "text" : 0.5, "graph" : 0.5, "aggregation": 0.5, "state": 1.0},
    {"name": " WordCloud", "text" : 0.5, "graph" : 0.5, "aggregation": 1.0, "state": 1.0},
    {"name": " TopicCloud", "text" : 0.5, "graph" : 0.5,  "aggregation": 1.0, "state": 1.0},
    {"name": " Ernährungspyramide", "text" : 0.8, "graph" : 0.2, "aggregation": 0.0, "state": 0.0}, #table
    {"name": " Ernährungspyramide", "text" : 0.1, "graph" : 0.9, "aggregation": 0.0, "state": 1.0}, #easy
    {"name": " Ernährungspyramide", "text" : 0.2, "graph" : 0.8, "aggregation": 0.5, "state": 1.0}, #mid
    {"name": " Ernährungspyramide", "text" : 0.3, "graph" : 0.7, "aggregation": 1.0, "state": 1.0}, #hard
    {"name": " Belly", "text" : 0.8, "graph" : 0.2, "aggregation": 0.0, "state": 0.0}, #table
    {"name": " Belly", "text" : 0.1, "graph" : 0.9, "aggregation": 0.5, "state": 1.0}, #pie
    {"name": " Belly", "text" : 0.0, "graph" : 1.0, "aggregation": 1.0, "state": 1.0}, #bar
    {"name": " Fulltext", "text" : 1.0, "graph" : 0.0,  "aggregation": 0.0, "state": 0.0},
    {"name": " Exploration 2", "text" : 0.9, "graph" : 0.1,  "aggregation": 0.0, "state": 0.0},
    {"name": " Theme Pyramid", "text" : 0.6, "graph" : 0.4,  "aggregation": 0.5, "state": 1.0},
    {"name": " HistoryCloud", "text" : 0.5, "graph" : 0.5,  "aggregation": 0.5, "state": 1.0},
    {"name": " Dashboard", "text" : 0.2, "graph" : 0.8,  "aggregation": 1.0, "state": 1.0},
    {"name": " Tilebar", "text" : 0.3, "graph" : 0.7,  "aggregation": 1.0, "state": 1.0},
    {"name": " Word Frequency", "text" : 0.3, "graph" : 0.7,  "aggregation": 0.0, "state": 1.0},
]
```

Listing 4.5: A static list of all implemented components with pre-defined values for the feature vector.

CHAPTER 5

Evaluation

This chapter shows the evaluation methodology and the results by matching functionalities of the implementation with information retrieval tasks in the knowledge visualization domain. These results are critically discussed, limitations are pointed out and possibilities for future work are proposed.

5.1 Evaluation Methodology

The evaluation is divided into two parts: The first part describes a scenario with 2 fictive personas with the intent of demonstrating functionality. The second part is a qualitative user study with 3 participants who have domain knowledge in the field. They received tasks and questions which they had to answer with the ET-extension, which are also reviewed and discussed.

5.1.1 Functionality Walkthrough

In this evaluation scenario we create 2 personas, each of them having knowledge about using web-applications, the functionalities of the implementation and the A^{+}CHIS system in general. The difference lies in the visualization literacy and component preferences of each persona. While persona 1 (P1) has high visual literacy and prefers complex visualizations (i.e., high aggregation), persona 2 (P2) prefers simpler visualizations (i.e., low aggregation). Figure 5.1 and 5.2 show the exploration process of the 2 exploration sessions respectively, with the black cards being components (FOCUS) and the arrows being navigation processes.

5.1.2 User Study

The user study in this project is designed to qualitatively evaluate the implementation based on exploration, tasks and questions which can be answered freely. The participants (T) are not

Figure 5.1: A sketch of the tracked/viewed components (including arrows for navigation) for the persona with high visualization literacy (P1).

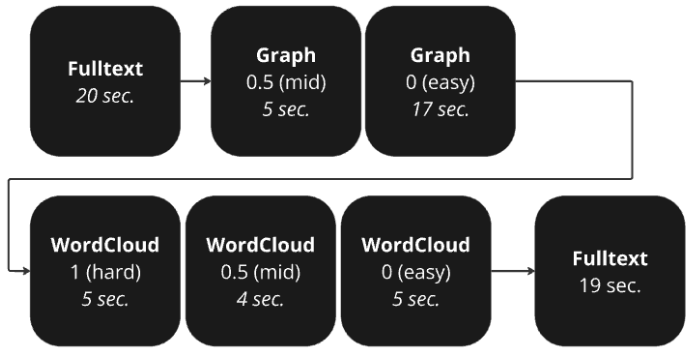


Figure 5.2: A sketch of the tracked/viewed components (including arrows for navigation) for the persona with low visualization literacy (P2).

familiar with this eye-tracking extension of A$^+$CHIS but all have a background in knowledge visualization. The focus here lies on discovering the usefulness of the main elements, detecting possible design flaws and receiving honest feedback over all.

The test setup consists of a laptop which is running A$^+$CHIS with the eyetracker and a mouse. The test procedure is as follows:

1. Setup of the test environment (without participant).
2. Greeting of the test person with explanation of the study procedure and the project.
3. Signing of the privacy statement (Appendix A.4).
4. Questionnaire (Appendix A.1)
5. Explanation of the setup and the most important controls/functions.
6. Free exploration by the user.
7. Completion of tasks (Appendix A.2) on a pre-recorded session (same for every partici-

pant), where the functions of the 3 main components are tested.

8. Interview with questions regarding the implementation (Appendix A.3).
9. Closing statements.

The whole procedure lasts 60 to 90 minutes. The timeline of the pre-recorded session (viewed components) is visualized in Figure 5.3.

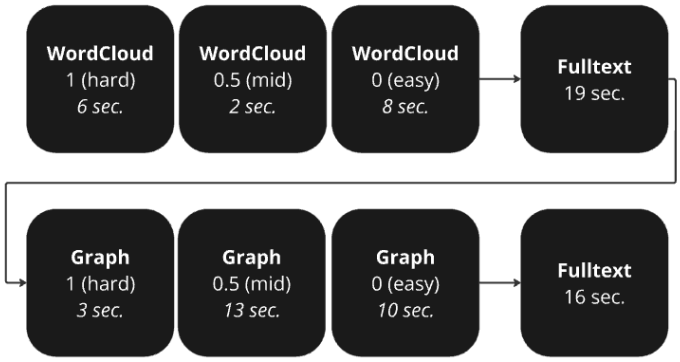
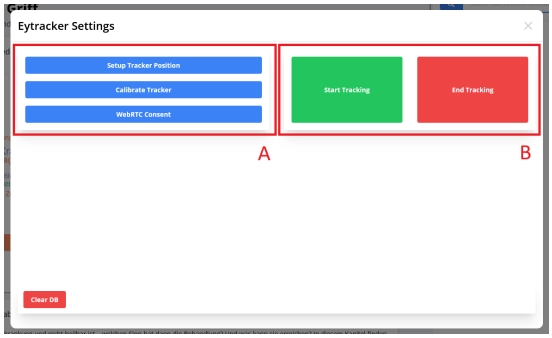


Figure 5.3: A sketch of the tracked/viewed components for the study tasks.

5.2 Results

The results are splitted up in the overall tracking process and the 3 implemented applications: (i) Timeline Analysis, (ii) Summative Analysis, and (iii) Recommendations.

After a short introduction into the functionalities, the ET-process was trouble-free for all study participants. Figure 5.4 (a) shows the implemented dashboard which is callable from all views of A$^+$CHIS and is used for setting up and activating the tracker. Figure 5.4 (b) shows a list for selecting the desired tracking session with options to delete or rename the instance.



(a) Panel for controlling the eye tracker. The buttons in blue (section **A**) control the settings, the buttons in section **B** are used to start or end the ET-session. *Clear DB*, which is intended for admin use only, deletes all database tables related to ET in the backend.

(b) List for selecting a session. The gray buttons allow editing the name of the session, the red ones delete the session, and the blue buttons are used to initiate the fixation computation and component matching, which results in displaying the 3 implemented applications.

Figure 5.4: The control panels for the ET-device (a) and session selection (b).

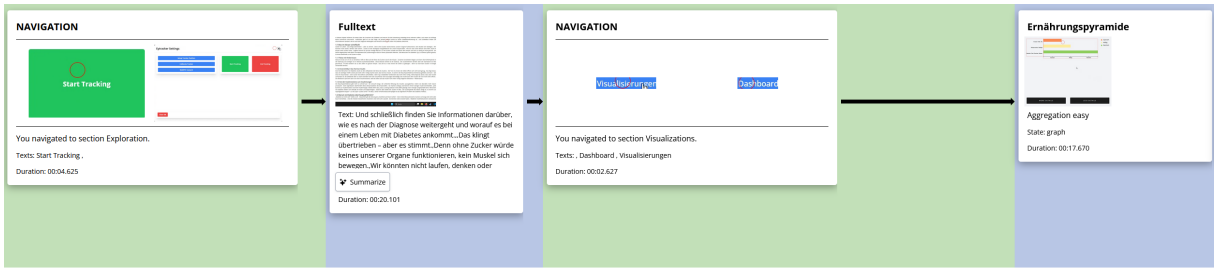
5.2.1 Timeline Analysis

Use Cases

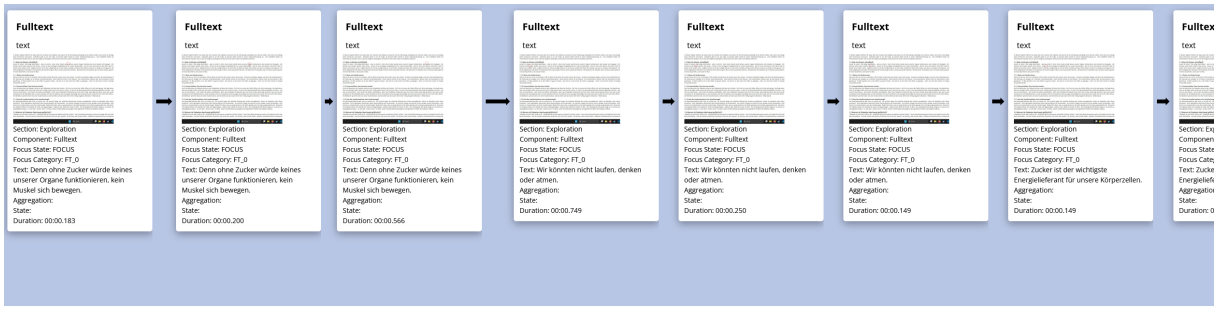
- Explainability by offering insight in the components used by a recommendation model.
- Recapturing gained knowledge for users.
- Retracing steps of a specific user for (study) evaluators.
- Compensating inaccuracies from the ET-device by summarization of texts.

Figure 5.5 (a) and (b) shows excerpts of the grouped- and non-grouped timeline for P2. Figure 5.5 (c) shows the hoverable gaze graph. Figure 5.6 (a) shows the actual gaze points from the tracking, while Figure 5.6 (b) visualizes the WordCloud summarization for T. The grouping is triggered by clicking a button on the left sidebar. From 6,622 initial gaze points recorded for P2, we compute 150 fixations and group them into 9 components and navigation entities. We observe similar results for P2 (6,113 $\rightarrow$ 191 $\rightarrow$ 7) and T (6,302 $\rightarrow$ 197 $\rightarrow$ 9), with all tracking sessions having a duration between 1 and 2 minutes.

During question-answering, all study participants found the grouping function to be one of the most useful features in the implementation. Two of them suggested to first show the grouped view and provide details on demand, according to *Shneiderman's Mantra* [46]. Regarding usability, two of the participants found the button for grouping to be too hard to find.

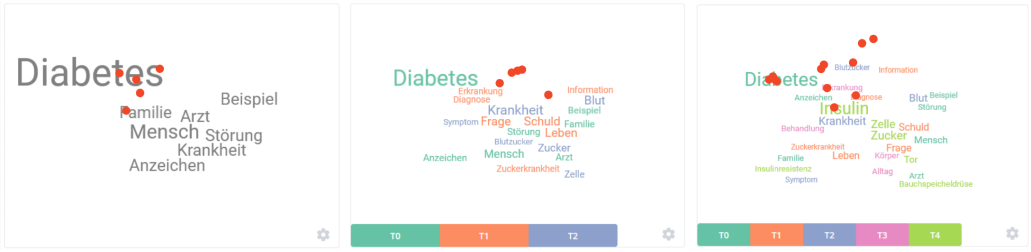


(a) P1 with grouping.

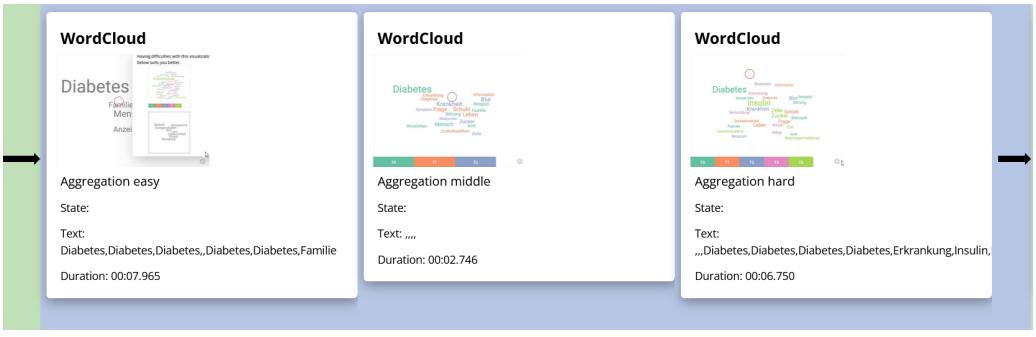


(b) P1 without grouping.

Figure 5.5: Results for the *Timeline Visualization*.



(a) Gaze Points in the *WordCloud* during tracking for our study participants.



(b) *WordCloud* summarization for our study participants.

Figure 5.6: *WordCloud* results for our study participants.

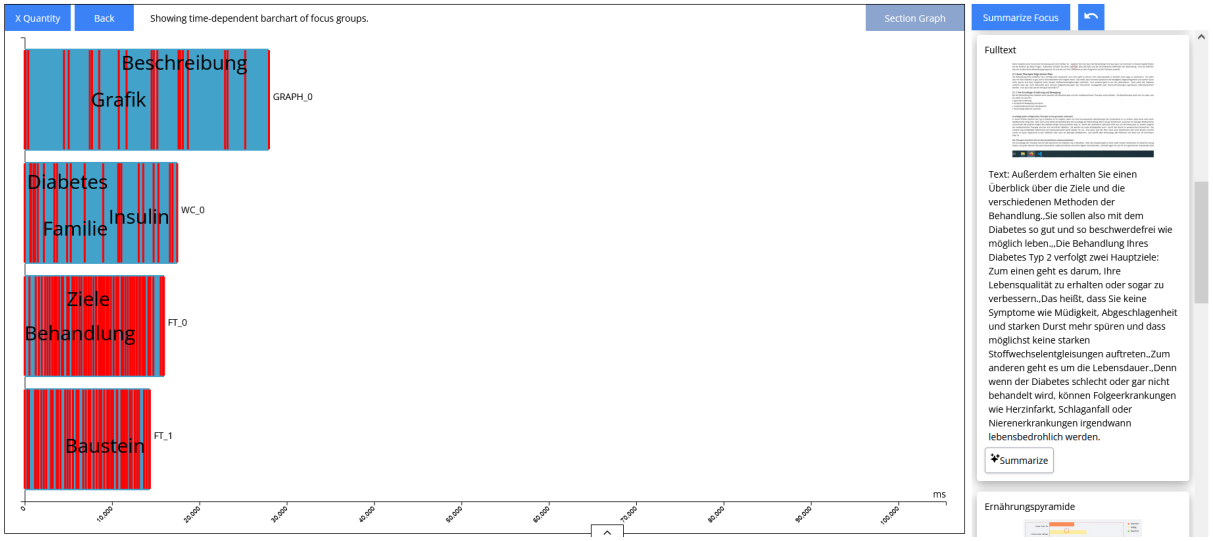
5.2.2 Summative Analysis

Use Cases

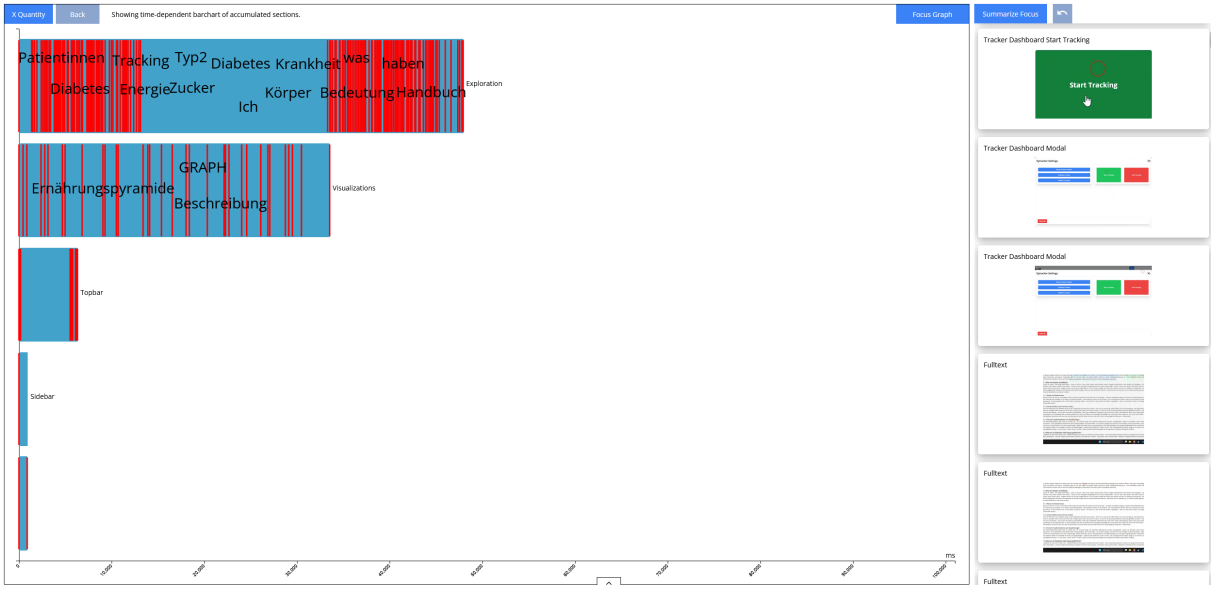
- Analysis support for (study) evaluators.
- Visualization and comparison of time spent in different components or parts of the application.
- Visualization of revisits for single words, areas and components.

Figure 5.7 shows results from the hierarchic knowledge collection: (a) displays the durations a user spent in components individually, which is the second step of our focus-based hierarchy (Focus State → **Focus Group** → Text/Position). The user in this visualization spent the most time in the first visit of the *Nutri Pyramid* (GRAPH_0). The study participants had difficulties finding this information, because the *Fulltext* component was viewed longer in total. This information can also be found, but in another hierarchy level; (b) shows the time the user has spent in different sections, being the first step of our section-based hierarchy (**Section** → Component → Text/Position).

Figure 5.8 shows the lowest hierarchy level (Text/Position) for the *Nutri Pyramid* (a) and *WordCloud* (b). As the horizontal lines define single fixations, we can observe (multiple) revisits on both sides and single visits of a word with longer durations in (b). This was a task (see Appendix 6, Tasks Sheet, Q6b) that lead to problems for the study participants, because the meaning of the lines was not clear to them.

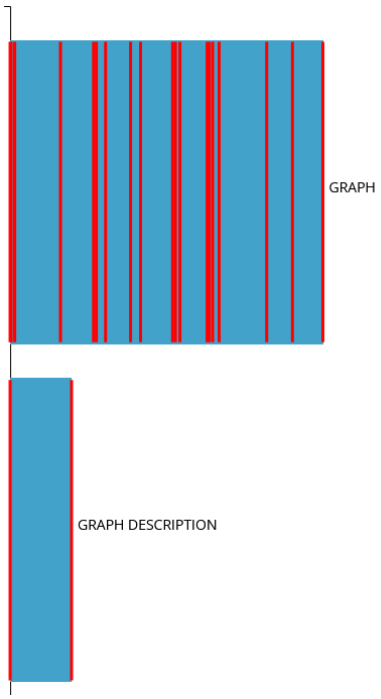


(a) Duration of elements in focus for the study participants.

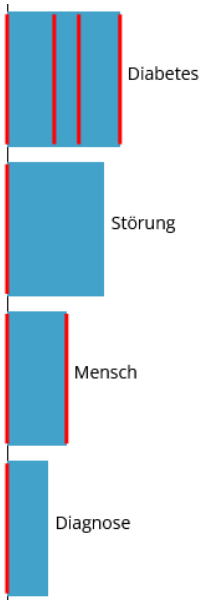


(b) Duration of sections in focus for study participants.

Figure 5.7: Results for the *Hierarchic Knowledge Collection* visualization.



(a) Duration for graph positions for study participants.



(b) Duration for WordCloud texts for P2.

Figure 5.8: Results for horizontal lines in the hierarchic knowledge collection visualization.

5.2.3 Recommendations

Use Cases

- Graph adaptation for the application based on FV value *aggregation*.
- Chapter recommendations based on similarity/dissimilarity of viewed content.
- Component recommendations based on high/low component preference.

Table 5.1 shows the results of the component- and content-based recommendations. The FV values indicate, according to the pre-defined interests of our personas, a higher interest in more advanced visualizations for P1 than for P2. Furthermore, as both looked at the same chapters, those recommendations are very similar.

The feedback from the study participants suggests that the most potential for future implementation lies in the recommendations section.

Table 5.1: Content- and Component-based recommendations for our persona and study sessions.

Session	FV ⟨text, graph, aggregation, state⟩	Recommended Chapters (similarity)	Recommended Components (aggregation)
Persona 1	$\langle 0.58, 0.42, 0.69, 0.59 \rangle$	1. 1.1 (62%) 2. 1.1.1 (55%) 3. 1(28%)	1. Theme Pyramid (0.5) 2. WordCloud (0.5) 3. HistoryCloud (0.5)
Persona 2	$\langle 0.52, 0.48, 0.08, 0.54 \rangle$	1. 1.1 (63%) 2. 1.1.1 (55%) 3. 1(35%)	1. WordCloud (0) 2. Word Frequency Graph (0) 3. WordCloud (0.5)
Study	$\langle 0.56, 0.43, 0.2, 0.53 \rangle$	1. 2.2 (56%) 2. 2.1 (55%) 3. 2(18%)	1. WordCloud (0) 2. Theme Pyramid (0.5) 3. WordCloud (0.5)

5.3 Discussion

The main challenge of this work lies in processing the natural movements of the eye and combining them with the display of structured information of a web application, which is a retrieval and filtering task. An implementation milestone is the creation of a fixation array, which can be seen as a solid basis for further usage in A$^+$CHIS or adaptive information systems in general.

Although the participants of our study reported usability issues, the 3 main tools (Timeline Analysis, Summative Analysis, Recommendations) show promising results in terms of application possibilities of the fixations array. Especially the summarization of the Timeline offers a clear overview of the exploration history, which should be displayed before showing individual fixations,in accordance with *Shneiderman's Mantra*. When comparing Figure 5.6 (a) and Figure 5.6 (b), we can also observe, that the text extraction from the fixations works as intended. The Hierarchic Knowledge Visualization offers comparability between viewing times and number of fixations for exploration specific areas, such as components or sections, which helps evaluators understand user behavior in the tracked application.

Table 5.1 shows, that our proposed system is able to offer meaningful component recommendations based on pre-defined and calculated features, as the calculated values for aggregation coincide with P1's and P2's (tracked) graph preferences. This means, that the provided data can be used for adaptation purposes in A$^+$CHIS. The content-based recommendations could

profit from further metrics, such as Exploration Coverage (see Chapter 3), which would allow for more targeted chapter proposals, potentially reducing confirmation bias.

5.4 Limitations and Future Work

The implemented system relies on tags in the HTML body. As manual tagging is a very time consuming task, a remedy in the future could be a Gen-AI based approach to automatically tag DOM objects, in order to improve the scalability of our implementation. This allows the consideration to not only run the it on A$^+$CHIS, but on a wide range of web-based applications with a similar framework.

Currently, one computational bottleneck is the image matching, which happens after computing fixations. The choice of *ffmpeg-python* lead to the design decision to take a screenshot and crop it inside a loop, with separately calling the input video for every fixation. One possibility to tackle this problem would be to call *FFMpeg* directly and make use of a "complex filter".

Although the accuracy of ET-devices is continuously improving, noise and head movements still have an impact on the tracking quality. High resolution displays have the ability to display small texts very well, which, on the other side, makes the task of extracting exact words and sentences harder. Although we counter this by using AI to summarize texts, the application would still profit from improved hardware.

The next step in implementation regarding the A$^+$CHIS project could be the implementation of an intelligent user interface with text and visualizations side-by-side, including our proposed ET implementation. In pre-defined intervals, the user interface shows the recommended visualizations and informs about interesting paragraphs to read. These recommendations can be improved by also implementing the discussed further metrics from Chapter 3 and evaluated in a larger scale to test for user acceptance and helpfulness.

CHAPTER 6

Conclusion

This thesis investigated the usage of eye tracking in the A$^+$CHIS document exploration system. The aim was to provide a system capable of tracking the users' gaze point, extracting relevant information from the interface as well as structuring and applying it. The findings demonstrate that we were able to offer an array of fixations which can be used for various visualizations and recommendations. We presented a timeline that can be summarized to give a clear overview of the exploration history (timeline analysis), a hierarchic knowledge visualization to compare exploration specific metrics (summative analysis) and content- and component-based recommendations, offering multiple adaptation possibilities for A$^+$CHIS. With developments, such as the introduction of Gen-AI based tagging and usability improvements, the implementation could be combined with a wide range of web-based applications to shape the future of adaptive visual exploration.

Bibliography

[1] O. K. Oyekoya and F. W. Stentiford, "Eye tracking: A new interface for visual exploration," *BT Technology Journal*, vol. 24, no. 3, pp. 57–66, 2006. DOI: 10.1007/978-0-387-76316-3_14. → [p1], [p3], [p6]

[2] N. J. S. Silva, *Adaptive User Interfaces Based on Visualization, Analysis and Prediction of User's Interactions and Behaviors*. PhD thesis, Graz University of Technology, 2020. → [p1]

[3] S. Egner, S. Reimann, R. Hoeger, and W. H. Zangemeister, "Attention and information acquisition: Comparison of mouse-click with eye-movement attention tracking," *Journal of Eye Movement Research*, vol. 11, no. 6, 2018. DOI: 10.16910/jemr.11.6.4. → [p1]

[4] T. Schreck, D. Albert, M. Bedek, K. Horvath, K. Jeitler, B. Kubicek, T. Semlitsch, L. Shao, and A. Siebenhofer-Kroitzsch, *Adaptive Visualization of Health Information Based on Cognitive Psychology - Scenarios, Concepts and Research Opportunities*, pp. 165–195. Springer International Publishing, 2023. DOI: 10.1007/978-3-031-34738-2_7. → [p3], [p6]

[5] R. Jianu, N. Silva, N. Rodrigues, T. Blascheck, T. Schreck, and D. Weiskopf, "Gaze-aware visualization: Design considerations and research agenda," 2025. (to appear). → [p3], [p5], [p9]

[6] A. T. Duchowski, *Eye tracking methodology: Theory and Practice*. Springer, 2017. DOI: 10.1007/978-1-84628-609-4. → [p3]

[7] S. Lukasser, "Document exploration based on eye tracking," 2022. Graz University of Technology. → [p4]

[8] C. Conati, G. Carenini, D. Toker, and S. Lallé, "Towards user-adaptive information visualization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015. DOI: 10.1609/aaai.v29i1.9775. → [p4]

[9] B. Steichen, G. Carenini, and C. Conati, "User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities," in *Proceedings of the 2013 international conference on Intelligent user interfaces*, pp. 317–328, Association for Computing Machinery, 2013. DOI: 10.1145/2449396.2449439. → [p4]

[10] J. A. Harsh, M. Campillo, C. Murray, C. Myers, J. Nguyen, and A. V. Maltese, ""seeing" data like an expert: An eye-tracking study using graphical data representations," *CBE—Life Sciences Education*, vol. 18, no. 3, p. ar32, 2019. DOI: 10.1187/cbe.18-06-0102. → [p4], [p23]

[11] J. M. Henderson and A. Hollingworth, "Chapter 12 - eye movements during scene viewing: An overview," in *Eye Guidance in Reading and Scene Perception*, pp. 269–293, Elsevier Science Ltd, 1998. DOI: 10.1016/B978-008043361-5/50013-4. → [p4]

[12] S. Xu, H. Jiang, and F. C. Lau, "Personalized online document, image and video recommendation via commodity eye-tracking," in *Proceedings of the 2008 ACM Conference on Recommender Systems*, pp. 83–90, Association for Computing Machinery, 2008. DOI: 10.1145/1454008.1454023. → [p4]

[13] R. Lengler, "Identifying the competencies of 'visual literacy' - a prerequisite for knowledge visualization," in *Tenth International Conference on Information Visualisation (IV'06)*, pp. 232–236, IEEE, 2006. DOI: 10.1109/IV.2006.60. → [p5]

[14] A. Locoro, W. P. Fisher, and L. Mari, "Visual information literacy: Definition, construct modeling and assessment," *IEEE Access*, vol. 9, pp. 71053–71071, 2021. DOI: 10.1109/ACCESS.2021.3078429. → [p5]

[15] M. Pachman, A. Arguel, L. Lockyer, G. Kennedy, and J. Lodge, "Eye tracking and early detection of confusion in digital learning environments: Proof of concept," *Australasian Journal of Educational Technology*, vol. 32, no. 6, 2016. DOI: 10.14742/ajet.3060. → [p5]

[16] P. R. DeLucia, D. Preddy, P. Derby, A. Tharanathan, and S. Putrevu, "Eye movement behavior during confusion: Toward a method," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 58, no. 1, pp. 1300–1304, 2014. DOI: 10.1177/1541931214581271. → [p5]

[17] A. C. Graesser, S. Lu, B. A. Olde, E. Cooper-Pye, and S. Whitten, "Question asking and eye tracking during cognitive disequilibrium: Comprehending illustrated texts on devices when the devices break down," *Memory & cognition*, vol. 33, no. 7, pp. 1235–1247, 2005. DOI: 10.3758/BF03193225. → [p5]

[18] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, *et al.*, "Knowledge graphs," *ACM Computing Surveys (Csur)*, vol. 54, no. 4, pp. 1–37, 2021. DOI: 10.1145/3447772. → [p5]

[19] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Trans. Graph.*, vol. 11, no. 1, pp. 92–99, 1992. DOI: 10.1145/102377.115768. → [p5]

[20] N. Kong, J. Heer, and M. Agrawala, "Perceptual guidelines for creating rectangular treemaps," *IEEE transactions on visualization and computer graphics*, vol. 16, no. 6, pp. 990–998, 2011. DOI: 10.1109/TVCG.2010.186. → [p5], [p6]

[21] T. Weber, H. Hußmann, and M. Eiband, "Quantifying the demand for explainability," in *Human-Computer Interaction – INTERACT 2021*, pp. 652–661, Springer International Publishing, 2021. DOI: 10.1007/978-3-030-85616-8_38. → [p6]

[22] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, and F. Herrera, "Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence," *Information Fusion*, vol. 99, p. 101805, 2023. DOI: 10.1016/j.inffus.2023.101805. → [p6]

[23] P. Lops, M. De Gemmis, and G. Semeraro, *Content-based Recommender Systems: State of the Art and Trends*, pp. 73–105. Springer US, 2011. DOI: 10.1007/978-0-387-85820-3_3. → [p6], [p7]

[24] C. Schwind and J. Buder, "Reducing confirmation bias and evaluation bias: When are preference-inconsistent recommendations effective – and when not?," *Computers in Human Behavior*, vol. 28, no. 6, pp. 2280–2290, 2012. DOI: 10.1016/j.chb.2012.06.035. → [p6]

[25] B. T. Carter and S. G. Luke, "Best practices in eye tracking research," *International Journal of Psychophysiology*, vol. 155, pp. 49–62, 2020. DOI: 10.1016/j.ijpsycho.2020.05.010. → [p9], [p10]

[26] D. D. Salvucci and J. H. Goldberg, "Identifying fixations and saccades in eye-tracking protocols," in *Proceedings of the 2000 symposium on Eye tracking research & applications*, pp. 71–78, Association for Computing Machinery, 2000. DOI: 10.1145/355017.355028. → [p10], [p11], [p31], [p32]

[27] A. Olsen, "The tobii I-VT fixation filter," *Tobii Technology*, vol. 21, pp. 4–19, 2012. → [p10]

[28] A. Olsen and R. Matos, "Identifying parameter values for an I-VT fixation filter suitable for handling data sampled with various sampling frequencies," in *proceedings of the symposium on Eye tracking research and applications*, pp. 317–320, 2012. → [p11]

[29] S. Kollmorgen and K. Holmqvist, "Automatically detecting reading in eye tracking data," vol. 144, pp. 1–9, 2007. → [p12]

[30] L. Shao, S. Lengauer, H. Miri, M. A. Bedek, B. Kubicek, C. Kupfer, M. Zangl, B. C. Dienstbier, K. Jeitler, C. Krenn, *et al.*, "Visual document exploration with adaptive level of detail: Design, implementation and evaluation in the health information domain," in *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2023, Volume 3: IVAPP, Lisbon, Portugal, February 19-21, 2023*, pp. 133–141, SciTePress - Science and Technology Publications, 2023. DOI: 10.5220/0011621800003417. → [p13]

[31] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl, "Visualization of eye tracking data: A taxonomy and survey," in *Computer Graphics Forum*, vol. 36, pp. 260–284, Wiley Online Library, 2017. DOI: 10.1111/cgf.13079. → [p16]

[32] S. Lee, S.-H. Kim, and B. C. Kwon, "Vlat: Development of a visualization literacy assessment test," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 551–560, 2017. DOI: 10.1109/TVCG.2016.2598920. → [p18]

[33] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972. DOI: 10.1108/eb026526. → [p20]

[34] R. Biedert, J. Hees, A. Dengel, and G. Buscher, "A robust realtime reading-skimming classifier," in *Proceedings of the symposium on eye tracking research and applications*, pp. 123–130, Association for Computing Machinery, 2012. DOI: 10.1145/2168556.2168575. → [p22]

[35] S. G. Eick, J. L. Steffen, E. E. Sumner, *et al.*, "Seesoft-a tool for visualizing line oriented software statistics," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 957–968, 1992. DOI: 10.1109/32.177365. → [p22]

[36] N. Jain, A. Bhansali, and D. Mehta, "Angularjs: A modern mvc framework in javascript," *Journal of Global Research in Computer Science*, vol. 5, no. 12, pp. 17–23, 2014. → [p26]

[37] B. Lesh, "Reactive Extensions for JavaScript." Version 7.5. Available: https://rxjs.dev/. → [p26]

[38] A. Wathan, "Tailwindcss." Version 3.1.4. Available: https://tailwindcss.com/. → [p26]

[39] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011. DOI: 10.1109/TVCG.2011.185. → [p26], [p35]

[40] Django Software Foundation, "Django." Version 3.2.9. Available: https://djangoproject.com/. → [p26]

[41] S. Tomar, "Converting video formats with ffmpeg," *Linux j.*, vol. 2006, no. 146, p. 10, 2006. → [p26], [p32]

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. DOI: 10.48550/arXiv.1201.0490. → [p26], [p36]

[43] OpenAI, "Chatgpt." Version GPT-4o mini, Jun 2024. Available: https://openai.com/. → [p26]

[44] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint*, 2023. DOI: 10.48550/arXiv.2302.11382. → [p26]

[45] P. Olsson, "Real-time and offline filters for eye tracking," 2007. Available: https://api.semanticscholar.org/CorpusID:14089141. → [p30], [p31]

[46] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *The Craft of Information Visualization*, Interactive Technologies, pp. 364–371, Morgan Kaufmann, 2003. DOI: 10.1016/B978-155860915-0/50046-9. → [p42]

List of Figures

List of Tables

List of Algorithms

List of Listings

List of Abbreviations

ADCS	Active Display Coordinate System
AI	Artificial Intelligence
AOI	Area of Interest
CHIS	Consumer Health Information System
DOM	Document Object Model
ET	Eye Tracking
FV	Feature Vector
I-VT	Velocity-Threshold Identification
LLM	Large Language Model
LoD	Level of Detail
ROI	Region of Interest
UCS	User Coordinate System
XAI	Explainable Artificial Intelligence

Appendices

CHAPTER A

Evaluation Documents

A.1 Evaluation Sheet

General Questions

Age: _____

What is your highest degree of education?

☐ secondary school
☐ bachelor's degree
☐ master's degree
☐ doctorate

What is your occupation at university?

☐ student
☐ student project assistant (A+CHIS)
☐ doctoral student

If you are a doctoral student, please state your field of research:

_____

Please rate your own domain experience in knowledge visualization from 1 (low) to 10 (high):

_____

A.2 Tasks Sheet

1. Navigate to “Eye-Tracking” and select the recording named “task-test”.

2. What was the first and the last Component, the user was exploring? (Excluding Navigation)

3. What was the first and the last word? (Excluding Navigation)

4. Identify the Component the user was exploring, before exploring “Ernährungspyramide”.

    a) What is this component called?

    b) Please summarize the text of this component (Write the first five words of the summarization and “…”)

5. Which difficulty/aggregation form did the user spend most time in, when looking at “WordCloud”?

6. Which Component did the user spend most time in in one go?

    a) Which graph area was most interesting for the user in that Component?

    b) Does the user have any patterns in “GRAPH DESCRIPTION”, that would indicate a revisit?

7. What aggregation form would the user get recommended, when using the A+CHIS application in production?

A.3 Specific Questions

1. How was your overall experience?

2. What parts of the application did you like?

   Why?

3. In your opinion, would an adaptive CHIS profit from the technologies offered in this work, in terms of adaptability?

   If yes, how?

4. In your opinion, would a user who wants to evaluate and recapture their knowledge profit from the technologies offered in this work?

   If yes, how?

5. Did you identify anything, that could be addressed in future work?

6. Did you identify anything that was confusing you?

A.4 Privacy Statement (German)

Datenschutzinformation

Projektinformationen
Arbeitstitel der Masterarbeit: Eye Tracking Support for Visual Exploration in a CHIS
Thema: Implementierung eines Eye Trackers in ein Livesystem, Extraktion relevanter Information und Erkundung von Möglichkeiten zur Unterstützung des Users beim Explorationsprozess.
Durchführende Person: Christoph Söls am Institut für Computergraphik und Wissensvisualisierung der Technischen Universität Graz

Welche personenbezogenen Daten werden erhoben und verarbeitet?
Im Rahmen der Forschungsarbeit werden folgende Daten verarbeitet:

- Alter, Bildungsstand, Beschäftigung, Erfahrung in der Wissensvisualisierung
- Interaktion mit dem A+CHIS-System (inkl. Screenrecording), z.B. Was wird angeklickt? Wie lange wird etwas betrachtet?
- Antworten auf spezifische Fragen

Was passiert mit den ermittelten Daten?
Die Daten werden zu Zwecken der Evaluierung der in der Masterarbeit dargestellten Anwendungen und Visualisierungen anonymisiert im Fließtext niedergeschrieben.
Eine gesonderte Speicherung personenbezogener Daten zu Zwecken des Nachweises der Einwilligung erfolgt nicht. Vielmehr ist eine Teilnahme an der Studie technisch nur bei vorheriger Erteilung der Einwilligung möglich.

Gesetzliche Grundlage für die Datenverarbeitung
Die Verarbeitung der Daten erfolgt aufgrund Ihrer Einwilligung (Art. 6 Abs 1 lit a DSGVO).

Wo und wie lange werden die Daten gespeichert?
Die Daten werden für die Dauer der Masterarbeit und darüber hinaus max. 7 Jahre aufbewahrt.

Informationen zu Ihren Rechten
Bezüglich Ihrer Betroffenenrechte auf Auskunft (Art. 15 DSGVO), Berichtigung (Art. 16 DSGVO), Löschung (Art. 17 DSGVO) und Einschränkung der Verarbeitung (Art. 18 DSGVO) wenden Sie sich an die unten angegebene Kontaktadresse der durchführenden Organisationseinheit oder die Datenschutzadresse der Universität. Falls Sie der Ansicht sind, dass wir mit unserer Verarbeitung gegen die DSGVO verstoßen, haben Sie ein Recht auf Beschwerde (Art. 77 DSGVO) bei der österreichischen Datenschutzbehörde, Barichgasse 40-42, 1030 Wien, Tel.: +43 1 52 152-0, E-Mail: dsb@dsb.gv.at.

Einwilligungserklärung

Wenn Sie an der Studie teilnehmen möchten, dann stimmen Sie bitte den folgenden Punkten zu:

- Ich willige ein, dass meine Daten, die im Zuge dieser Studie in anonymisierter Form erhoben werden, für Forschungszwecke verarbeitet und zur Veröffentlichung wissenschaftlicher Arbeiten in anonymisierter Form verwendet werden.
- Ich stimme zu, dass meine Daten nach vollständiger Anonymisierung zum Zweck der wissenschaftlichen Transparenz in offenen Datenbanken verfügbar gemacht werden.
- Ich kann meine Einwilligung zur Verarbeitung meiner Daten jederzeit zurückziehen. Durch den Widerruf wird die Rechtmäßigkeit, der bis zum Widerruf verarbeiteten Daten nicht berührt.
- Ich habe die obigen Datenschutzinformation gelesen und erkläre mich zur Teilnahme bereit. Ich weiß, dass ich die Teilnahme jederzeit ohne Nennung von Gründen abbrechen kann.

Unterschrift und Datum: _____