Visual Analysis of Sequential Data

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik sowie dem Stuttgarter Zentrum für
Simulationswissenschaft der Universität Stuttgart
zur Erlangung der Würde einer
Doktorin der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von

Tanja Munz-Körner,
geb. Munz

aus Waiblingen

Hauptberichter:	Prof. Dr. Daniel Weiskopf
Mitberichter:	Univ.-Prof. Mag. Dr. Silvia Miksch
Tag der mündlichen Prüfung:	18. Juli 2025

Visualisierungsinstitut
der Universität Stuttgart

2025

Acknowledgments

I would like to thank everyone who worked with me during my doctoral research, especially the people from VISUS and VIS. A special thanks goes to my supervisor, Daniel Weiskopf, for giving me the opportunity to work on my doctoral thesis. He supported me with his guidance, experience, and constructive feedback during this time. I am very grateful that he was my supervisor. I would also like to thank Silvia Miksch for reviewing my dissertation and taking part in my PhD defense. I am very glad that Michael Burch supervised both my student and diploma theses, offered me the opportunity to work as a student assistant, and continued to show interest in my projects even after I graduated. Without that time and our joint publications, I would not have started with this adventure. A big thank you goes to my co-authors: Fabian Beck, Tanja Blascheck, Michael Burch, Lewis Chuang, Rafael Garcia, Frank Heyen, Sebastian Künzel, Kuno Kurzhals, Paul Kuznecov, Michael Neumann, Daniel Ortega, Sebastian Pannasch, Yoeri Poels, Noel Schäfer, Michael Sedlmair, Pascal Tilli, Toon van Benthem, Dirk Väth, Sandeep Vidyapu, Ngoc Thang Vu, and Eugene Zhang as well as his entire family, especially Andrew and Steve. I would also like to thank all the people I have not yet mentioned and with whom I shared an office, as well as those who co-supervised student projects with me: Manuel Álvarez Chaves, Anneli Guthke, David Hägele, Daniel Klötzl, Maurice Koch, Marius Kurz, Antoine Lhuillier, Julian Lißner, Nils Rodrigues, and Christoph Schulz. Finally, I would like to thank all the other colleagues I have worked with at VIS(US).

Lastly, I thank my family, who supported me through every phase of my doctoral journey. Without their support, this work would not have been possible. I want to thank my husband Peter, my daughter Cecilia – who was born during the final phase of my dissertation and has since deeply influenced my life – and my parents, who always supported me in every way they could.

My work at the Visualization Research Center of the University of Stuttgart was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – TRR 161 (project B01) – 251654672, and under Germany's Excellence Strategy – EXC 2075 – 390740016.

Contents

List of Figures

Chapter 4

Chapter 5

List of Tables

List of Abbreviations

AI	artificial intelligence
ANN	artificial neural network
AOI	area of interest
BERT	Bidirectional Encoder Representations from Transformers
BLEU	bilingual evaluation understudy
BPE	byte pair encoding
CharacTER	Translation Edit Rate on Character Level
CNN	convolutional neural network
CV	computer vision
DE-EN	German-to-English
DL	deep learning
DNN	deep neural network
EN-DE	English-to-German
EOS	end-of-sequence
EP	expected prediction
ETFuse	Eye Tracking Fusion System
FNN	feedforward neural network
GAT	graph attention network
GNN	graph neural network
GPT	Generative Pre-trained Transformer
GRU	gated recurrent unit
Hz	Hertz
I-DT	dispersion-threshold identification
I-VT	velocity-threshold fixation identification
IMDb	Internet Movie Database
LSTM	long short-term memory
MDS	multidimensional scaling
ML	machine learning
ML4Vis4ML	machine learning for visualization for machine learning
MLP	multilayer perceptron
NLP	natural language processing
NLTK	Natural Language Toolkit
NMT	neural machine translation
NMTVis	Neural Machine Translation Visualization System
NN	neural network
PCA	principal component analysis

ppd pixels per degree

px pixels

ReLU rectified linear unit

RMSE root mean squared error

RNN recurrent neural network

Seq2Seq sequence-to-sequence

SMACOF Scaling by MAjorizing a COmplicated Function

SOS start-of-sequence

SUS System Usability Scale

SVM support vector machine

t-SNE t-distributed stochastic neighbor embedding

UMAP uniform manifold approximation and projection

Vis4ML visualization for machine learning

VisME Visual Microsaccades Explorer

VQA visual question answering

XAI explainable artificial intelligence

Abstract

Sequential and temporal data is omnipresent in various areas of our lives. It is characterized by a sequence of data points in a fixed order, possibly with a temporal component. With an increasing amount of data being generated and collected, and different types of data originating from various domains, appropriate methods are needed to examine, interpret, understand, and draw conclusions from complex processes. Depending on the use case, the amount of data, and the target group, different analysis methods have to be chosen or developed. While visualization alone can already provide interesting insights into the data, interactive visual analysis helps users extract additional information by letting them focus on specific parts of the data and exploring it from different perspectives. Techniques such as brushing and linking and multiple coordinated views (multiple visualizations for the same data that are linked) help realize such an examination.

In this thesis, several approaches for visually analyzing sequential data are presented. The focus lies particularly on two key application areas: eye tracking and the interpretability of machine learning (ML) methods. Additionally, the use of dimensionality reduction methods during preprocessing for visualization is an important concept of this work. In all these areas, sequential or temporal components play important roles. They can be the subject of exploration, used as input data to trigger complex processes, represent internal mechanisms within methods, or be the output of a process. Users may want to examine or compare them to understand the data better. In the area of eye tracking analysis, this thesis presents a visual analysis approach that addresses the influence of various filter settings (parameter choices) on the data being visualized and interpreted. Additionally, a method is presented that combines temporal data from different sources to enable a better comparison of this data. Preprocessing steps play a crucial role in both methods to allow meaningful visualizations of the data and subsequent examination of the data. Next, various ML approaches are considered. The interpretability of ML techniques is currently a very important and challenging topic. Especially ML models in the area of natural language processing (NLP) deal with sequential components as input data, and also, the internal operations follow sequential processing steps. This thesis demonstrates that, in the field of NLP, internal information from neural machine translation (NMT), visual question answering (VQA), and text classification tasks can be made available to users for an enhanced understanding of internal mechanisms and to improve prediction results. Toward the end of this thesis, dimensionality reduction techniques are applied as a preparation step

for visualizing sequential data. First, dimensionality reduction is used in an interactive system to examine text classification in the context of ML. However, interpreting 2D visualizations of dimensionally reduced sequential data requires careful consideration due to the possibility of data loss, misleading projections, and potential misinterpretation of the visualization itself. Therefore, in this work, visualization approaches are presented that address this challenge to provide methods to prevent misinterpretation. Overall, all presented interactive visualization approaches of this thesis use sequential data as input, and the visual analysis techniques help users during data exploration, interpretation, for debugging purposes, or to improve prediction results generated with ML models.

Zusammenfassung

Sequenzielle und zeitliche Daten sind in verschiedenen Bereichen unseres Lebens allgegenwärtig. Sie sind durch eine Aneinanderreihung von Datenpunkten mit festgelegter Reihenfolge gekennzeichnet, gegebenenfalls mit einer zeitlichen Komponente. Mit einer zunehmenden Menge an erzeugten und gesammelten Daten sowie den unterschiedlichen Datenarten aus verschiedenen Bereichen werden geeignete Methoden benötigt, um komplexe Vorgänge zu untersuchen, zu interpretieren, zu verstehen und Schlussfolgerungen daraus zu ziehen. Je nach Anwendungsfall, Datenmenge und Zielgruppe müssen unterschiedliche Analysemethoden gewählt oder entwickelt werden. Während eine Visualisierung bereits interessante Einblicke in die Daten gewähren kann, hilft die interaktive visuelle Analyse den Nutzern, zusätzliche Informationen zu extrahieren, indem sie sich auf bestimmte Bereiche der Daten konzentrieren und diese aus verschiedenen Blickwinkeln untersuchen können. Techniken wie das Auswählen und Verknüpfen (Brushing and Linking) und mehrere koordinierte Ansichten (mehrere Visualisierungen für dieselben Daten, die miteinander verknüpft sind) helfen bei der Umsetzung einer solchen Untersuchung.

In dieser Arbeit werden verschiedene Ansätze zur visuellen Analyse sequenzieller Daten vorgestellt. Der Fokus liegt insbesondere auf zwei zentralen Anwendungsbereichen: Blickpunktverfolgung (Eye-Tracking) und Interpretierbarkeit maschineller Lernmethoden. Darüber hinaus ist die Verwendung von Methoden zur Dimensionsreduzierung während der Vorverarbeitung zur Visualisierung ein wichtiges Konzept dieser Arbeit. In all diesen Bereichen spielen sequenzielle oder zeitliche Komponenten eine wichtige Rolle. Sie können Gegenstand der Untersuchung sein, als Eingabedaten für die Auslösung komplexer Prozesse dienen, interne Mechanismen innerhalb von Methoden darstellen oder die Ausgabe eines Prozesses sein. Benutzer möchten sie möglicherweise untersuchen oder vergleichen, um die Daten besser zu verstehen. Im Bereich der Eye-Tracking-Analyse wird in dieser Arbeit ein visueller Analyseansatz vorgestellt, der sich mit den Auswirkungen verschiedener Filtereinstellungen (Parameterauswahl) auf die visualisierten und zu interpretierenden Daten befasst. Zusätzlich wird eine Methode vorgestellt, die zeitliche Daten aus verschiedenen Quellen kombiniert, um einen besseren Vergleich dieser Daten zu ermöglichen. Bei beiden Methoden spielen Vorverarbeitungsschritte eine entscheidende Rolle, um eine sinnvolle Visualisierung der Daten und eine anschließende Untersuchung zu ermöglichen. Anschließend werden verschiedene maschinelle Lernansätze betrachtet. Hier ist die Interpretierbarkeit von Methoden des maschinellen Lernens derzeit ein sehr wichtiges und herausforderndes Thema. Insbesondere Modelle

des maschinellen Lernens im Bereich der natürlichen Sprachverarbeitung (NLP) beschäftigen sich mit sequenziellen Komponenten als Eingabedaten und auch die internen Vorgänge folgen sequenziellen Verarbeitungsschritten. In dieser Arbeit wird für den Bereich NLP gezeigt, dass interne Informationen aus neuronaler maschineller Übersetzung (NMT), visuellen Frage-Antwort-Systemen (VQA) und Textklassifizierungsaufgaben den Nutzern zum besseren Verständnis interner Vorgänge und zur Verbesserung von Vorhersageergebnissen zur Verfügung gestellt werden können. Gegen Ende dieser Arbeit werden Techniken zur Dimensionsreduktion als Vorbereitungsschritt für die Visualisierung sequenzieller Daten angewendet. Zunächst wird Dimensionsreduktion in einem interaktiven System zur Untersuchung der Textklassifizierung im Kontext des maschinellen Lernens eingesetzt. Die Interpretation von 2D-Visualisierungen dimensionsreduzierter sequenzieller Daten erfordert jedoch eine kritische Betrachtung aufgrund der Möglichkeit eines Datenverlusts, missverständlichen Projektionen und möglicher Fehlinterpretation der Visualisierung selbst. Daher werden in dieser Arbeit Visualisierungsansätze vorgestellt, die diese Herausforderung angehen und eine Möglichkeit bieten, Fehlinterpretationen zu vermeiden. Zusammengefasst nutzen alle vorgestellten interaktiven Visualisierungsmethoden dieser Arbeit sequenzielle Daten als Eingabe, und die visuellen Analysetechniken helfen Benutzern Daten zu analysieren, zu interpretieren, Fehler zu suchen und zu beseitigen oder Vorhersageergebnisse zu verbessern, die mit Methoden des maschinellen Lernens erzeugt wurden.

CHAPTER 1

Introduction

Many measurements, mechanisms, and processes have a sequential or temporal component. Analyzing them in different application domains is essential to understanding and interpreting them, as well as to improving the quality of underlying or related procedures. The selection of an analysis method is crucial for data interpretation and to gain insights into the data. Visual analysis processes are especially valuable in analyzing such data. For this thesis, different visual analysis approaches were developed and evaluated, focusing on interactive visualization to facilitate human-machine collaboration. Two research areas are primarily examined and used as application examples: eye tracking and the interpretability of machine learning (ML) approaches with sequential aspects.

With eye tracking [158], human gaze can be recorded at a given frequency over time to analyze where they look. Visual analysis helps explore gaze positions through different interactive visualizations. Here, the raw gaze positions or extracted eye movements, consisting of an aggregation of multiple gaze positions, can be used as sequences or time series in the analysis. When looking at a visual stimulus, humans show individual gaze behaviors with different patterns, trends, and possibly anomalies in their eye movement. Various types of eye movements can be extracted and analyzed from the recorded gaze positions. In a visual analysis approach, raw data and extracted information can be used to analyze gaze patterns and their influence on the attention and perception of one or multiple people. Additionally, it is, for example, possible to evaluate underlying mechanisms for the extraction of eye movements.

The second application domain used in this thesis is ML. The interpretability of ML, especially deep learning (DL), is currently an important and growing research area, also known as explainable artificial intelligence (XAI). Results generated with ML approaches are often astonishing but sometimes still wrong or unexpected. The models are usually hard to understand and gain insights into. They are frequently considered black boxes, where internal mechanisms and results created during prediction are difficult to grasp. One goal in this context is to open the black box, making ML models more transparent and controllable via visual analysis. This thesis explores and evaluates visual analysis methods for ML in the context of several concrete examples from natural language processing (NLP). In ML, different sequential or temporal components can be found varying from the input data and the training process to the internal states during prediction and possibly the output. When data from NLP is used, the analysis of text components plays an important role. Text consists of individual words that are ordered and processed in a sequential way. Internal mechanisms such as the training or prediction process are often performed in a temporal manner. Especially the visual analysis of internal states of such ML models helps make ML models more interpretable and steerable. Then, it is possible to better understand prediction results and their relation to internal states. Additionally, it may help in debugging and data curation, hence improving models or prediction results.

Various aspects of sequential data can be explored through visual analysis, especially through interactive data exploration with visualizations. Here, the potential of computer-generated visualizations and the ability of users to interactively explore, understand, interpret, and modify them can be leveraged.

Before a user can actually explore data through visualization, the input data must be preprocessed to allow the generation of visualizations and subsequent analysis. Such data preparation and transformation includes, for example, filtering, joining, aggregating, and the extraction of important features. This step plays an essential role in all projects presented in this thesis. It is, for example, also a part of the visualization pipeline as proposed by Card et al. [76]. In particular, the use of dimensionality reduction techniques to reduce the number of features in the data for visualization will be examined in detail.

This thesis presents different approaches with the common goal of providing mechanisms for visual analysis of (multidimensional) sequential data to interactively analyze the data or related processes. In all projects, preprocessing is of substantial importance. In the analysis of gaze data, the influence of filter settings for detecting eye movements and the possibilities of joining data from different devices is explored. In the context of XAI [50], neural machine translation (NMT) [297, 305], visual question answering (VQA) [41], and the

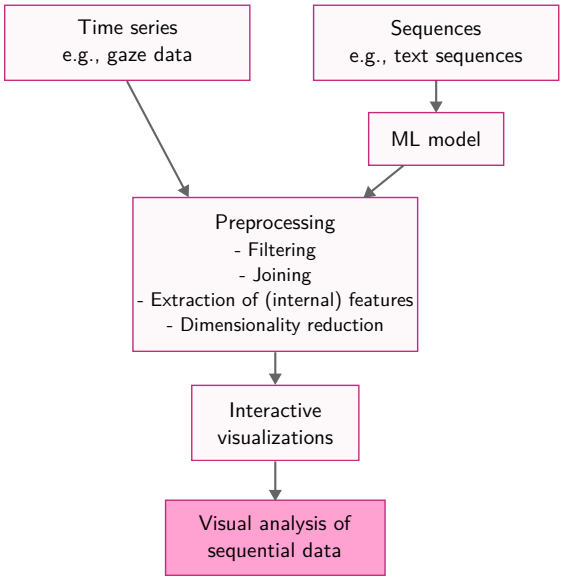


Figure 1.1 — Generalized workflow for the proposed visual analysis approaches in this thesis: from input data (sequences or time series, e.g., in the form of gaze data or text), over preprocessing (such as filtering, joining, extraction of information, or dimensionality reduction) and interactive visualizations (usually multiple connected views with brushing and linking) to the visual analysis. Note that the importance of the individual steps within the approaches presented in this thesis varies. In addition, some of the presented methods contain additional elements such as loopbacks for generating improved models and newly created input data, or the models include additional (non-sequential) input data (e.g., images).

classification of text sequences are investigated. Corresponding ML models require the possibility to extract internal information generated during the prediction for visualization. Finally, commonly used unsupervised ML methods are employed as a preparation step for the 2D visualization of sequential data, and improved visualization methods are presented to avoid misinterpretation.

Figure 1.1 provides an overview of the different aspects this thesis covers with respect to the visual analysis of sequential data: from the sequential or temporal input, the preprocessing, and interactive visualizations to the analysis. Each step of this workflow plays a key role in the visual analysis processes presented in this thesis. Often, approaches must be specifically designed for the individual input data type, analysis goals, and target groups to be able to handle all important aspects of the data. Preprocessing steps are crucial for extracting essential information, aggregating or filtering the data, or general preparation for the visualization on a 2D plane. Finally, the actual visualizations with possibilities for interaction must be selected or designed such that the relevant information can be presented meaningfully to users and that the target group of the approaches can easily understand and work with the presented visualizations.

Terminology

In this thesis, the terms *projection* and *to project* are used to describe the process of mapping high-dimensional data into a lower-dimensional space (e.g., 2D) using dimensionality reduction techniques. While some of these techniques, such as PCA [160], satisfy the mathematical definition of a projection, others do not (e.g., t-distributed stochastic neighbor embedding (t-SNE) [319]), since they do not have properties such as linearity and idempotence. However, for consistency, the term *projection* is used throughout this thesis for all dimensionality reduction methods.

1.1 Research Questions

Different research questions were formulated and examined throughout this thesis in the context of the research areas mentioned above.

- The first research question deals with the visual analysis of sequential and temporal data. Visual analysis approaches can help users explore and better understand data with different structures and from various application domains. This thesis focuses on sequential data and the usefulness of developed visual analysis approaches.

  > **RQ1**: *How beneficial is interactive visual analysis in exploring sequential data?*

  Many visualization and visual analysis approaches already exist for sequential and temporal data in different domains [311]. The focus of this thesis lies particularly on eye tracking data and data generated with sequence-based ML approaches. Each data type and corresponding analysis goal may require a specifically designed visual analysis solution. Then, different evaluation methods can be employed to verify the usefulness of the approaches. These methods can include user studies, computer-based evaluations, and the demonstration of use cases.

- Many algorithms provide (hyper)parameters to allow users access to different variables used while processing some data. (Hyper)parameters are variables (i.e., values or settings) used in an algorithm during execution. In the context of ML, they are usually called hyperparameters. These parameters have a strong impact on the behavior of an algorithm, its performance, its output, and therefore also on the analysis results. Many research studies do not even report on their parameters when they present their results in

publications. In the context of this thesis, we analyze their influence on the input data used for visualization and the analysis.

> **RQ2**: *How does the choice of (hyper)parameter values (in preprocessing and the analysis) affect the visual analysis of sequential data?*

When parameter values are chosen poorly, the visualizations and the conclusions drawn from the visualized data can be misleading or even incorrect. Therefore, it is often important to select these parameter values carefully and to explore their influence by comparing results with differently chosen values.

- The third research question deals with the interpretability of sequential or temporal components in ML. ML, and especially DL [31], are currently important research areas where understanding internal processes is crucial. Especially, the internal states of a model are interesting during the exploration and might be used to draw conclusions about the final prediction result.

> **RQ3**: *How can visual analysis of internal (sequential) components of ML methods improve their interpretability?*

Many systems and approaches exist in the context of XAI with the goal of increasing the interpretability of ML models through visualizations [351]. Nevertheless, there are still many open challenges when it comes to providing users with a system to understand their ML models and explain the prediction results. The designs of these systems depend, for example, on the input data, selected model, target group, and analysis goals. Since it is often necessary to include information from the specific input data of a given task into the system, the analysis solutions may not always be generalizable to data with other specifications or from other domains. Additionally, there exists a large amount of different ML architectures with different internal mechanisms that may require individual visualizations. This thesis focuses on the field of NLP that uses sequences (and possibly other additional data formats) as input, different sequential (internal) mechanisms during the prediction, and possibly sequences as output. While users can have access to specific internal structures, it is still relevant if this information is helpful to users during the analysis. Additionally, resources might need specific preparation to make them accessible to users, e.g., through adaption of the architecture of the ML model or through transformation for visualization.

- Finally, the last research question deals with dimensionality reduction as a means to explore sequential and temporal data. Using dimensionality reduction techniques, high-dimensional data can be mapped to a lower dimensional space (e.g., two dimensions), and the newly generated data can be used for visualization. Such a mapping can be applied to a variety of different data types. In the literature, using this technique for static data with multiple dimensions to generate and analyze 2D visualizations is common. Usually, the visualizations consist of dots, each representing a data instance. Here, distances between dots and clusters of dots can be used to relate to the structure of the original data. In previously published work, this technique was also used for temporal data (see Table 5.1 on page 171). In addition to dots, instances are connected to show the temporal order. However, none of the existing literature explored if such an approach is actually appropriate for the interpretability of patterns in temporal multidimensional data in 2D and if projection problems and new visual elements may be problematic for the interpretation. Therefore, the usefulness of such utilization should be explored.

  RQ4: *How appropriate are dimensionality reduction techniques for visually exploring sequential data?*

  In this thesis, temporal components of ML models are extracted and projected to 2D for the visual analysis of the underlying structure of internal information. Additionally, issues related to such a visualization technique are discussed, and visualization methods are proposed to avoid misinterpretation of the data. Artificial data, simulations, and observation results exemplify these new methods.

The individual research questions do not stand for themselves but are interconnected. All projects presented in this thesis relate to at least two of these research questions, most of them to even more. An overview of the individual visual analysis approaches presented in this thesis and their relation to related mechanisms and the research questions can be found in Figure 1.2.

1.2 Outline and Contribution

This section provides a summary of each chapter of this thesis and outlines my contribution to published papers, partially reused in this thesis, and the corresponding source code available for all projects. Additional material (such as videos, input data, or trained models) is also available for some projects. My supervisor, Daniel Weiskopf, was involved in all the methods presented in this

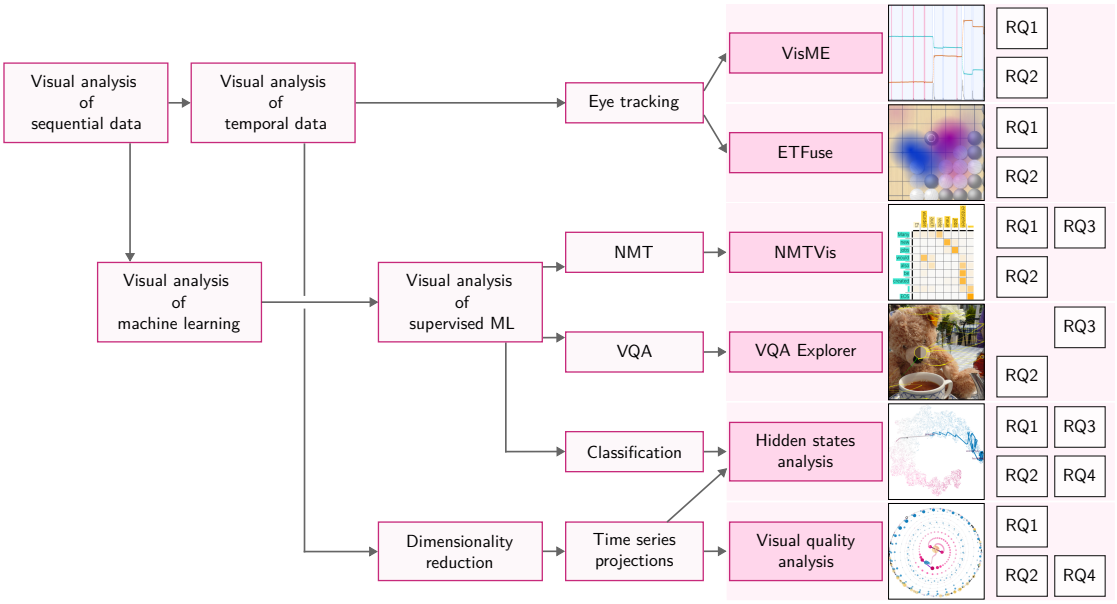


Figure 1.2 — Overview of the different visual analysis approaches presented in this thesis: On the right side, the different visual analysis systems and the respective research questions they address are listed in the order in which they are presented in this thesis. The left side of the diagram shows the thematic context of the individual visual analysis approaches.

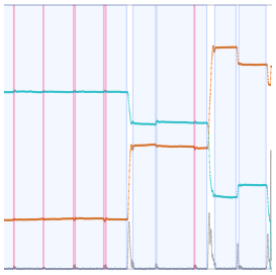
thesis, proposing ideas and continuously giving support and feedback. He is a co-author of each publication.

After providing background information in Chapter 2, three chapters present different types of visual analysis approaches for sequential data: Chapter 3 focuses on techniques that rely on temporal data, specifically on the analysis of eye tracking data. Chapter 4 explores examples from NLP, where text sequences are used as input, and visualizations in the context of XAI are presented. Finally, in Chapter 5, dimensionality reduction techniques are employed to prepare data for 2D visualization. The examples used in this chapter contain both sequential and temporal data: data in the context of NLP and other areas such as climate data. This chapter additionally explores the visualization method itself and how to improve it to avoid misinterpretation. To conclude this work, Chapter 6 contains a summary and overarching discussion for this thesis.

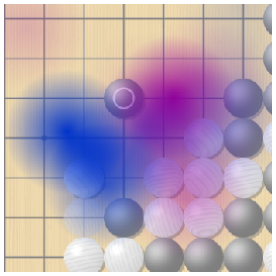
Chapter 2 – Background In Chapter 2, background information about different important aspects of this thesis is provided. This includes the topic of visual analysis, the differences between sequential and temporal data, the main application domains explored in this thesis (eye tracking and ML), background on

determining requirements throughout the projects, and the employed evaluation methods.

Chapter 3 – Visual Analysis of Temporal Eye Tracking Data Afterward, in Chapter 3, two visual analysis methods for temporal data are presented, especially dealing with RQ1 and RQ2. The proposed approaches can be used to process and analyze eye tracking data.



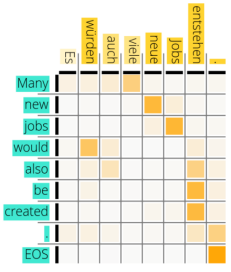
The first work (Section 3.1) presents a visual analytics approach to explore microsaccade distributions in high-frequency eye tracking data. To detect microsaccades automatically, eye movement filter algorithms are commonly used. However, it is observed that different studies may employ different parameter values even when utilizing the same algorithms. These parameter values describe important properties of microsaccades such as the amplitude or duration. To address this issue and ensure reproducibility in the analysis of microsaccade data, a visual analytics system called Visual Microsaccades Explorer (VisME) is introduced. The goal of this work was to create an interactive visualization system to explore microsaccades in the context of the eye tracking data. I implemented the system, performed the evaluation with use cases and a user study, and wrote most of the paper. Daniel Weiskopf and Lewis Chuang (from LMU Munich) supervised this project, gave advice, and revised the final paper. Sebastian Pannasch (from Technische Universität Dresden) provided one of the datasets and gave final feedback on the work. This work was published in the *Journal of Eye Movement Research (JEMR)* [10]. The source code is publicly available via Zenodo [8] and on GitHub (`https://github.com/MunzT/VisME`).



The second approach presented in Chapter 3 (Section 3.2) is a visual analysis system for eye movement data from two people playing virtual board games (i.e., online games) against each other or in collaboration. The approach offers different methods to synchronize and align gaze recordings and mouse click events from two eye tracking setups. Analysts can then visually examine the joined data using a combination of techniques, including attention maps, gaze plots, and a temporal summary that displays the distance between gaze positions and shows temporal positions of mouse events. The idea for this work was by Eugene Zhang and Kuno Kurzhals. The project started as a bachelor's thesis by Noel Schäfer [279], supervised by Tanja Blascheck, Kuno Kurzhals, Eugene Zhang (from Oregon State University), and me. Noel Schäfer wrote the initial source code. Together, we first created a demo paper for *ACM Symposium on*

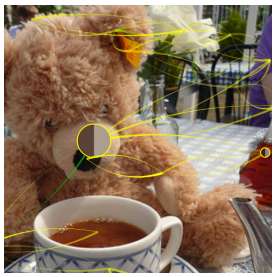
Eye Tracking Research and Applications (ETRA) [13] and later an extension as a full paper for the *International Symposium on Visual Information Communication and Interaction (VINCI '20)* [12]. I coordinated this project during the writing phase, made some changes and extensions to the source code written in Java, and created all visualizations for the papers and videos demonstrating our work. Additionally, I wrote most of the text for the paper. Kuno Kurzhals was responsible for the related work section, and Eugene Zhang strongly contributed to the evaluation and its interpretation. Every author substantially revised the final paper. I prepared and delivered the online presentation for the conference. The source code is publicly available on DaRUS [11] and GitHub (`https://github.com/MunzT/ETFuse`).

Chapter 4 – Visual Analysis of Deep Learning with Sequential Aspects In Chapter 4, two analysis methods for NLP applications are presented: the first focuses on the interpretability of NMT, and the second on VQA. Here, the focus lies on RQ3 and partially on RQ1 and RQ2.



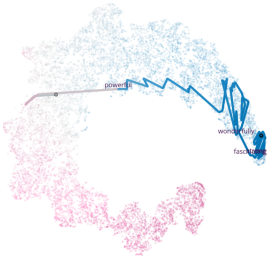
The first part of this chapter (Section 4.1) introduces a novel interactive method for visually analyzing, comprehending, and improving NMT. The system assists users in automatically translating documents using NMT while also identifying and correcting potential translation errors. The corrections made by users can then be utilized to refine the NMT model, which can lead to automatic improvements in the entire document. This project was done in collaboration with the Institute for Natural Language Processing (IMS) at the University of Stuttgart and started as Paul Kuznecov's master's thesis, supervised by Thang Vu and me. Paul Kuznecov created the base implementation for this approach. Later, Dirk Väth joined the project. He was primarily responsible for maintaining the ML-related areas of the project (e.g., adding the Transformer architecture to the system and writing related parts in the paper). I coordinated this project during the writing phase, implemented additional features and visualizations for the system, improved its usability, and wrote most of the paper. This also included the creation of all images for the paper and videos for demonstration. Dirk Väth, Thang Vu, and Daniel Weiskopf contributed to polishing this work. The initial paper [15] was published at *Graphics Interface 2021* and was selected as one of the top submissions. We were invited to submit a revised and extended version of the article to the journal *Computers & Graphics* [19]. I prepared and delivered the online presentation at the *Graphics Interface* conference. The source code for the analysis systems of both paper versions [16, 18] is available on

DaRUS and GitHub (https://github.com/MunzT/NMTVis). The trained models are also available on DaRUS [17].

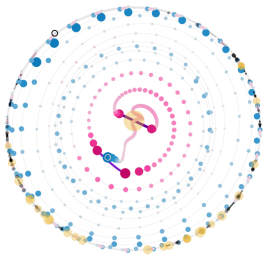


Then, a new visual analysis system is presented to enhance the interpretability of scene-graph-based VQA (Section 4.2). This approach focuses on supporting users in detecting and improving issues with the prediction due to poor scene graphs. Additionally, with the help of interactive visualizations, users gain insights into the model's decision-making processes. Pascal Tilli (from the IMS at the University of Stuttgart) and I devised the idea for this project. We formulated goals and ideas for the approach. Together with Sandeep Vidyapu, we supervised the master's thesis of Noel Schäfer [280]. Noel Schäfer implemented the first version of our approach with some support from Pascal Tilli and wrote most parts of our *International Symposium on Visual Information Communication and Interaction (VINCI '23)* paper [25]. During the writing phase of the paper, Sebastian Künzel joined the project, supported the supervision, wrote some paragraphs, and created the visualizations and a video with Noel's tool. I substantially revised the paper, wrote most of the related work section, and gave advice. The scope of the contributions to this work of Sebastian Künzel, Pascal Tilli, and me was equal. Sebastian Künzel prepared and delivered the presentation at the conference. We were invited to submit an extended version of our *VINCI* paper to the journal *Visual Computing for Industry, Biomedicine, and Art (VCIBA)* [7]. Sebastian Künzel, Pascal Tilli, and I contributed equally to this publication with a different focus. While Sebastian Künzel was mainly responsible for adding new features to the implementation and for the user study, Pascal Tilli added details about the underlying mechanisms of the VQA system. I was responsible for the use cases and describing additional features of our interface. Each author substantially revised the manuscript. The source code of this project is available on DaRUS [26, 28] and GitHub (https://github.com/Noeliel/GraphVQA-Explorer). Additionally, some supporting material is provided on DaRus [27]. I helped with both the documentation and publication of these sources.

Chapter 5 – Projection-based Visual Analysis of Sequential and Temporal Data In Chapter 5, projection-based analysis of sequential data is used in the context of ML (hidden states as sequential data). Additionally, the limitations of such a method for multidimensional time series projections are presented together with new visualization methods for better interpretability. This chapter deals especially with RQ3 and RQ4, but also RQ1 and RQ2.



First, a visual analytics approach is presented for ML experts who wish to analyze hidden states of layers within recurrent neural networks (RNNs) (see Section 5.1). This technique enables users to interactively examine how hidden states store and process information during the propagation of the input sequence through the network. By utilizing this method, users can gain insights into the prediction mechanism. For example, they can determine which parts of the input data have a greater influence on predictions and understand how the model associates a specific output with a configuration of a hidden state. This work is an extension of the work by Garcia and Weiskopf [129]. I used the available material from the previous project (source code of Jupyter Notebooks that were used for the creation of visualizations in this previous work) and created an interactive visualization system based on Python 3, JavaScript, and D3.js for the visualizations. I incorporated additional visualizations, a variety of options for adjusting the visualizations and used color throughout different visualizations more consistently. I updated and extended the original paper to describe all new features adequately and added new examples. This includes the creation of all illustrations in the paper and a video with the help of the interactive system. Rafael Garcia and Daniel Weiskopf gave advice and helped edit the final version of the paper. The corresponding paper [5] was published in the journal *Visual Computing for Industry, Biomedicine, and Art (VCIBA)*. The source code is publicly available on DaRUS [14] and GitHub (`https://github.com/MunzT/hiddenStatesVis`). For the *International Conference on Data-Integrated Simulation Science (SimTech2023)*, I prepared a poster [24] with supplemental material on DaRUS [23]. This poster is also based on this work. Additionally, it contains content from Munz et al. [15, 19], which is presented in Section 4.1 of this thesis. I summarized these works and highlighted their differences regarding the visual analysis approaches for ML. Sebastian Künzel and Daniel Weiskopf gave feedback. I presented the poster at the conference.



While the previous approach uses dimensionality reduction to map hidden states to 2D for visualization, the second approach in Chapter 5 (Section 5.2) focuses more on the interpretability of such a technique. The focus lies on addressing the challenge of visualizing multidimensional time series data by projecting it into 2D space. New techniques are introduced to handle and display projection errors that may arise during the process of projection and visualization. To represent the temporal nature of the data, successive time instances are depicted as dots connected by lines or curves, indicating their temporal dependencies.

However, inaccurately projected data points, inconsistent variations in distances between projected time instances, and intersections of connecting lines can lead to incorrect interpretations of the original data. To mitigate these issues, novel methods are proposed to address and resolve these challenges effectively. Daniel Weiskopf introduced me to the idea of this project. Afterward, its direction was refined multiple times. I created the source code, and most of the paper was written by me. Additionally, I selected, prepared, and analyzed the examples. Daniel Weiskopf continuously gave feedback and helped finalize the publication. The source code is available on DaRUS [20] and GitHub (`https://github.com/MunzT/visualQuality`), and some supplemental material is available on DaRus [21].

Chapter 6 – Conclusion The last chapter includes a summary and a final conclusion with an overarching discussion as well as ideas for future research directions. Especially, the initially stated research questions are answered in the context of the presented approaches.

Copyright

In this thesis, material from several publications is reused. This is done under their respective copyrights and with the kind permission of all co-authors:

- Material from Munz et al. [10] is published by *Bern Open Publishing* in the special thematic issue: *Microsaccades: Empirical Research and Methodological Advances*. The work is licensed by the *Journal of Eye Movement Research* under the Creative Commons Attribution 4.0 International License and the copyright is held by the authors. It is allowed to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon it for any purpose) the material.
- Material from Munz et al. [12] is published by the *Association for Computing Machinery (ACM)* in the proceedings of the *13th International Symposium on Visual Information Communication and Interaction (VINCI '20)* and the copyright is held by the authors. The owners have the right to reuse any portion of the work, without fee, in any future works written or edited by the author, including books, lectures and presentations in any and all media.
- Material from Munz et al. [13] is published by *ACM* in the *ETRA '20 Adjunct Proceedings: Symposium on Eye Tracking Research and Applications Proceedings* and the copyright is held by the authors.
- Material from Munz et al. [15] is published by *Graphics Interface – Canadian Information Processing Society*, and the copyright is held by the authors.

Permission is granted to *CHCCS/SCDHM* to publish in print and digital form, and ACM to publish electronically.

- Material from Munz et al. [19] is published by *Elsevier Ltd* in the *Computers & Graphics* journal and the copyright is held by *Elsevier Ltd*. The authors are granted (without the need to obtain further permission) the *Author Rights*, including among others (e.g., personal use and internal institutional use) the use of the manuscript by the authors in a subsequent compilation of the author's works or re-use by the author of portions or excerpts in other works (with full acknowledgment of the original publication of the Article).
- Material from Schäfer et al. [25] is published by *ACM* in the proceedings of the *16th International Symposium on Visual Information Communication and Interaction (VINCI '23)* and the copyright is held by the authors.
- Material from Künzel et al. [7] is published by *Springer* in the *Visual Computing for Industry, Biomedicine, and Art (VCIBA)* journal, licensed under the Creative Commons Attribution 4.0 International License, and the authors hold the copyright. It is allowed to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon it for any purpose) the material.
- Material from Garcia et al. [5] is published by *Springer* in the *Visual Computing for Industry, Biomedicine, and Art (VCIBA)* journal, licensed under the Creative Commons Attribution 4.0 International License, and the authors hold the copyright. It is allowed to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon it for any purpose) the material.
- Material from Munz et al. [24] is published by *University of Stuttgart* for the *International Conference on Data-Integrated Simulation Science (SimTech2023)* and the authors hold the copyright.
- Material from Munz and Weiskopf [22] is published by *Elsevier B.V.* on behalf of *Zhejiang University and Zhejiang University Press Co. Ltd* in the *Visual Informatics* journal, licensed by the publisher under the Creative Commons Attribution 4.0 International License, and the authors hold the copyright. It is allowed to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon it for any purpose) the material.

Additional Contribution

During the time I worked on this thesis, I contributed to further publications, but their content is not part of this work:

The approach to generate an overlap-free version of generalized Pythagoras trees for hierarchy visualization [9] is based on a project I worked on as a graduate student, and that was published afterward [1, 2]. The extension was done in collaboration with the Eindhoven University of Technology and the University of Duisburg-Essen. Toon van Benthem and Yoeri Poels did a student project supervised by Michael Burch where they implemented an overlap-free version of the original work. Fabian Beck asked me to join the project during the writing phase of the paper. I was responsible for creating the visualizations with the students' system and implemented a different color scheme to make the visualizations look more appealing. Additionally, I helped finalize the paper. This work was published as a short paper in the proceedings of the *IEEE Visualization Conference (VIS)*. I presented this work at the corresponding conference in Vancouver. Daniel Weiskopf gave advice on the paper and revised the final version of the paper and presentation. The source code, for which I implemented the current color scheme and created the documentation, is available on GitHub (`https://github.com/Benthem/FDGPT`).

Another work introducing an interactive visual comparison system for classifiers started as the master's thesis of Frank Heyen [153]. Michael Sedlmair, who supervised this work with me, developed the original idea. Frank Heyen did most of the work, ranging from implementing the visualization system and performing user studies to writing most of the publication. Michael Neumann and Daniel Ortega from the IMS at the University of Stuttgart contributed to the user study. Throughout the master's thesis and when writing the paper, I contributed with advice. Additionally, I revised the paper. The source code is available on GitHub (`https://github.com/fheyen/ClaVis`).

Supervised Student Projects

I supervised multiple student projects during my research. Some of them were used as a base for subsequent publications, where some of the content can also be found in this thesis. These are the student projects I (co-)supervised that took place during this time. Daniel Weiskopf was the examiner for each thesis if not mentioned otherwise.

- Master's thesis by Paul Kuznecov [196]: "A visual analytics approach for explainability of deep neural networks." Co-supervised by Ngoc Thang Vu. Examined by Daniel Weiskopf and Ngoc Thang Vu. This thesis was the base for the approach by Munz et al. [15, 19], which is also presented in Section 4.1 of this thesis.
- Bachelor's thesis by Jamie Ullerich [317]: "Stimulus generation software for the analysis of smooth pursuits." Co-supervised by Antoine Lhuillier.

- Bachelor's thesis by Noel Schäfer [279]: "Visuelle Eye-Tracking-Analyse bei kooperativen Spielszenarien." Co-supervised by Tanja Blascheck, Kuno Kurzhals, and Eugene Zhang. This thesis was the base for the approach by Munz et al. [12, 13], which is also presented in Section 3.2 of this thesis.
- Master's thesis by Frank Heyen [153]: "Visual parameter space analysis for classification models." Co-supervised by Michael Sedlmair and Ngoc Thang Vu. Examined by Michael Sedlmair. This thesis was the base for the approach by Heyen et al. [6].
- Master's thesis by Shashank Ramesh Salian [275]: "Deep visualization for MR-based biological age estimation." Co-supervised by Karim Armanious and Sherif Abdullatif.
- Bachelor's research project by Daniel Bin Schmid, Pascal Walloner, and Satoaki Eitschberger [277]: "A curated collection of multivariate time series for visualization benchmarks." Co-supervised by David Hägele.
- Bachelor's thesis by Muhammed Kaya [181]: "System zur visuellen Analyse für Daten der kontinuierlichen Glukosemessung von Diabetes-Patienten."
- Master's thesis by Noel Schäfer [280]: "Visual Analytics für Visual-Reasoning-Aufgaben." Co-supervised by Pascal Tilli and Sandeep Vidyapu. This thesis was the base for the approach by Schäfer et al. [25], which is also presented in Section 4.2 of this thesis.
- Master's thesis by Satish Aanand Balasubramanian [48]: "Visual analysis of a machine learning approach applied to turbulence data." Co-supervised by Daniel Klötzl and Marius Kurz.
- Master's thesis by Ahmed Hassan [144]: "Visualization of neural networks for diagnostic hydrological modeling." Co-supervised by Anneli Guthke, Nils Rodrigues, and Manuel Álvarez Chaves.
- Master's thesis by Hai Dang Nguyen [234]: "Visual exploration for deep learning models and trainings for microstructure data." Co-supervised by David Hägele and Julian Lißner.

CHAPTER 2

Background

This thesis aims to present several visual analysis approaches. Different data is employed, with a focus on sequential data: especially data from eye tracking and output from ML models. Additionally, dimensionality reduction techniques are used to create visualizations for temporal data. This chapter provides an overview of relevant background information on related topics, including information on visual analysis, characteristics of sequential data, a description of eye tracking and supervised/unsupervised ML, details on determining the requirements, and an overview of evaluation methods.

This section contains content from the following publications:

- S. Künzel, T. Munz-Körner, P. Tilli, N. Schäfer, S. Vidyapu, N. T. Vu, and D. Weiskopf. Visual explainable artificial intelligence for graph-based visual question answering and scene graph curation. *Visual Computing for Industry, Biomedicine, and Art (VCIBA)*, 8(1):9, 2025, doi: 10.1186/s42492-025-00185-y [7].
- T. Munz-Körner and D. Weiskopf. Exploring visual quality of multidimensional time series projections. *Visual Informatics*, 8(2):27–42, 2024, doi: 10.1016/j.visinf.2024.04.004 [22].
- T. Munz-Körner, S. Künzel, and D. Weiskopf. Poster: Visual-explainable AI: The use case of language models. In *International Conference on Data-Integrated Simulation Science (SimTech2023)*, 2023, URL: https://www.simtech2023.uni-stuttgart.de/documents/Theme-2/Munz-Koerner-Tanja.pdf [24].
- T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. Visualization-based improvement of neural machine translation. *Computers & Graphics*, 103:45–60, 2022, doi: 10.1016/j.cag.2021.12.003 [19].
- T. Munz, L. L. Chuang, S. Pannasch, and D. Weiskopf. VisME: Visual microsaccades explorer. *Journal of Eye Movement Research*, 12(6), 2019, doi: 10.16910/jemr.12.6.5 [10].

2.1 Visualization and Visual Analysis

Information visualization is a great means to explore data and gain insights. With graphical representations of the data, information is easily accessible to humans for their interpretation. Such visualizations enable efficient data exploration and help identify patterns and outliers. Typical visualizations include charts, scatterplots, histograms, heatmaps, trees, graphs, and maps. Information visualization specifically focuses on the preparation and visualization of abstract data.

Visual analysis is the process of examining, understanding, and interpreting data using visual representations. The goal is to gain insights into the oftentimes complex, large, and possibly multidimensional data. Visual analysis helps users discover patterns, trends, anomalies, and hidden relationships. Visual analytics [85, 182, 308] is a related concept. Here, a combination of interaction techniques, computational resources, and human abilities, such as perception and cognition, are combined to explore the data. Through the help of interactive tools and techniques, users are supported in interacting with and adapting the visualizations. Interaction and filtering techniques can, for example, help users link corresponding data elements in different views that represent the same data or change the type and amount of information currently being visualized. This allows users to perform different tasks with the underlying data, understand it better, and answer various questions about the data. The terms *visual analysis* and *visual analytics* are also used interchangeably to relate to

the concept of analyzing and extracting insights from visual data representations. Both approaches can involve interactive visualizations and analysis techniques. Throughout this thesis, these terms are not strictly separated. Their usage in the different chapters primarily relates to the use of terms in the related publications.

The book edited by Thomas and Cook [308] is centered around visual analytics. It contains the following description: "*People use visual analytics tools and techniques to synthesize information and derive insight from massive, dynamic, ambiguous, and often conflicting data; detect the expected and discover the unexpected; provide timely, defensible, and understandable assessments; and communicate assessment effectively for action.*"

In their work [308], different focus areas are defined:

- *Analytical reasoning techniques* play an important role, especially in gaining insights and decision-making. In this thesis, this is a crucial aspect, for example, when debugging a complex ML system or correcting translation results.
- The *visual representations and interaction techniques* are crucial to exploring the data. Interaction techniques, in addition to visualizations, are valuable to explore and understand a considerable amount of data. In this work, different interaction techniques are used to allow users to analyze the data in various ways and from different perspectives. Often, different input data requires specific interactive visualization-based approaches for meaningful exploration. In the following chapters, several methods are introduced.
- *Data representations and transformations* are essential for the actual data visualization. Data may be preprocessed through transformations to be visualized in a reasonable and accessible way. This thesis uses techniques such as extracting important information, joining data from different sources, filtering for subsets of the data, and projecting data with dimensionality reduction techniques. Such preprocessing is a key step involved throughout this thesis. Different preprocessing steps are used for each visualization system presented in the following chapters, to enable meaningful visualizations.
- Methods for the *production, presentation, and dissemination* of analysis results are essential to share the final results with others as images, videos, or integrated into presentation slides. The visualizations created by the systems presented in this thesis can be used for this purpose.

Overall, information visualization, visual analytics [182, 308], and the visual analysis process in general, can be advantageous approaches in developing

visual exploration systems across various domains. Adding interaction techniques for visualizations is crucial for complex data analysis and for supporting the analysis process. Users can extract significant information and perform a detailed exploration of the data through interaction and filtering. A key technique is here the use of multiple visualizations in coordinated views [273]. This means that different aspects of the data can be shown simultaneously, and changes may affect all visualizations at the same time. With the help of brushing and linking [332], users may select elements in one of the visualizations, and all additional visualizations show corresponding information or updates. This technique helps better understand connections between the data. Interactive filtering enables the dynamic adjustment of the type and quantity of data used for the visual representations in real-time. Often, making interactive adjustments to settings can modify how the visuals are presented and perceived. Tooltips play a crucial role in connecting quantitative and visual information by offering additional numerical or textual details. They can be shown on demand when the mouse cursor is moved over certain visual elements. This supports users gain a deeper understanding of the visual elements. Generally, combining such techniques helps better explore and understand the data.

2.2 Sequential and Temporal Data

Sequential data can be characterized as a sequence of successive observations where the order of data points is crucial. Such data is available in many domains of different sizes and complexities. Temporal or dynamic data is a subset of sequential data. Here, timestamps specify the order of the data. Temporal data includes, for example, weather data, eye tracking data, the training process in ML, or any other data that is provided with timestamps. A time series is a typical example of temporal data where data points are usually recorded at a specific time interval, for example, every second. A sentence or a whole text document is an example of sequential data without timestamps but where the order of words is essential. When processing text, this process can also create a time series, e.g., when the next word is always processed in the next time step. The previous description mentioned that there is a difference between sequential data and its subset, temporal data. However, these two terms are sometimes also used interchangeably without clear differentiation. It is also important to note that the term time series can also refer to a sequence without explicit timestamps, particularly when the focus is on the sequential process in which the data is being used.

While sequential data usually consists of one sequence of successive data elements in a row, it is also possible that several branches represent different

alternative paths, similar to the branching in a tree. Additionally, the various branches may later merge again.

The dimensionality of sequential data can vary depending on its origins and influence how it can be explored and visualized. While temperature observations for one location consist of only one dimension, weather observations with different measurements for one location usually consist of multiple dimensions. When not just one location is explored but a large area or multiple different places, many dimensions are available at each time step, leading to even high-dimensional data.

In the context of this thesis, sequential and temporal data primarily used include data from eye tracking (considered as time series) and text sequences (processed as sequences) in the context of ML. More details about these two domains follow in Section 2.4 and Section 2.5. Besides these data types, further sequential information is being analyzed in addition (such as events occurring at specific times (Section 3.2)). This work also uses some data from other sources: video clips (earth observations) and simulation data (see Section 5.2). To explore complex behaviors, it is sometimes not enough to look at only the sequential components of a method and ignore further data sources. Therefore, non-sequential data is used as input in this thesis as well, e.g. images (Section 3.1) and scene graphs (Section 4.2).

Different aspects are essential in analyzing sequential data: Beyond the dimensionality and origins of the data, the target groups and analysis goals play a crucial role in the choice of visualization design and interaction techniques.

2.3 Visual Analysis of Sequential Data

Sequential data, particularly temporal data, exists all around us, and its exploration is highly significant. Therefore numerous approaches exist for the visualization of temporal data [32, 33, 121, 311] and event sequences in general [140].

The simplest form of sequential or temporal data is one-dimensional. This means a series of values is given in a specific order, possibly with timestamps. A line chart is a straightforward visualization technique where the order or time is plotted on one axis and numerical values on the other. For multidimensional data, multiple curves can be displayed in the same diagram or as small multiples adjacent to each other. Stacked area charts [302] are an extension of line charts showing multiple curves where the individual magnitudes are stacked on top of each other. This way, it is also possible to see aggregated values at each step. The areas in between are filled in different colors or patterns. ThemeRivers [145]

and horizon graphs [147] are an extension hereof. Instead of using curves to visualize the values of instances, it is also possible to use bar charts. Here, the height of the bars represents the values, and it is possible to stack multiple bars on top of each other or plot them adjacent to each other. While this method can be used for individual time steps, it is also helpful for aggregated intervals. If categorical data has to be presented, matrices are an option. Again, time is plotted on one axis and the category or categories along the other one, potentially using properties like color for additional dimensions (see Section 5.1 for an example). Besides using a rectangular coordinate system, the data can instead, for example, be plotted circularly [357] or on a spiral [335]. This can be applied to curves, bar charts, matrices, or other representation forms, and it may be helpful in identifying circular patterns. Additionally, there are techniques that were initially not designed for sequential data, but they are also applicable for creating visualizations for such data. For example, tree representations can be employed for branching over time to show possible alternative paths (e.g., as used by Strobelt et al. [301] and in this thesis in Section 4.1 for the beam search visualization). Graphs (e.g., as used by Collins et al. [92]) are used when the data merges again later in time.

If the sequential data has a spatial component, locations can be plotted on a 2D plane where lines connect individual instances. This can be used to show geographical locations on a map or in eye tracking. For example, a scanpath or gaze plot [238, 239, 347] (see Chapter 3) can depict the eye movement over time on a screen. If we extend such an approach, we can even create 3D plots where the order or time is mapped to an additional axis. An example is a space-time cube [44]. However, additional interaction techniques to rotate the cube might be necessary due to occlusion.

Sequential data can also be represented through animation [49]. This is intuitive for temporal data since time is mapped to time. However, it might be challenging to follow the temporal changes and compare the data at different times.

If the exact sequential development is not of interest but instead summarized statistical information about all data samples, different types of charts can be used. This includes, for example, histograms and scatterplots. We use this method in Section 3.1 to show aggregated temporal information for specified time periods.

Creating a meaningful visualization might sometimes be challenging since data can be multi- or high-dimensional. Here, dimensionality reduction techniques are a frequently employed method to reduce the dimensionality of the input data. With dimensionality reduction (e.g., principal component analysis (PCA)

or t-SNE), multidimensional time series data can be mapped to 2D or 3D for visualization and analysis [35, 45]. In such visualizations, the data samples are represented as dots that are connected by lines to show their sequential order. This approach may reveal important characteristics and patterns of the underlying data [45]. However, when projecting data to a lower-dimensional space, it is necessary to consider that not all important information can be maintained, and projection errors and artifacts can be introduced. Additionally, the interpretation of the resulting visualizations might be challenging. In Chapter 5 we use such an approach for the visualization of the temporal development; Section 5.2.1 contains more details about this approach.

Many more visualization and visual analysis techniques exist for sequential data. These include combinations and extensions of existing approaches [173], or specifically designed new visualizations. Interactive systems or dashboards, in particular, allow a combination of different visualization types in coordinated multiple views [273] and support visual analysis through interaction techniques such as brushing and linking [332], filtering, aggregated information, and details on demand. This enables the exploration of larger datasets with many dimensions and a study from different perspectives. Often, specific visual analysis approaches are developed for individual use cases by combining existing visualization methods. An example is the approach by Burch et al. [4] for time-varying weighted digraphs with a hierarchical structure. Multiple different visualizations are used: a layered icicle plot [188] as hierarchy visualization, a matrix visualization for the temporal development of edge weights, and parallel stacked lines for edges that summarize information over time. Similarly, developer rivers [3] use a visualization as a combination of ThemeRivers [145] and Sankey diagrams [315] (a visualization showing the direction and magnitude of flows) in an interactive visualization approach for the analysis of software development processes. The visual analysis approaches proposed in this thesis also use a combination of different visualization methods and extend them for the specific use cases.

The data domain and target groups for visual analysis systems play an additional significant role in determining the design and functionality of such systems: Domain experts have extensive knowledge and expertise about the specific type of data and may have different requirements for visualizations and interaction techniques than non-experts. Experts may require more advanced tools and functionalities to perform tasks like debugging and conducting intense data analysis. Additionally, their knowledge allows them to understand complex visualizations and to use advanced features for in-depth explorations. Non-experts do not possess the same level of specialized knowledge but are still interested in exploring the data. Therefore, they may have other goals in

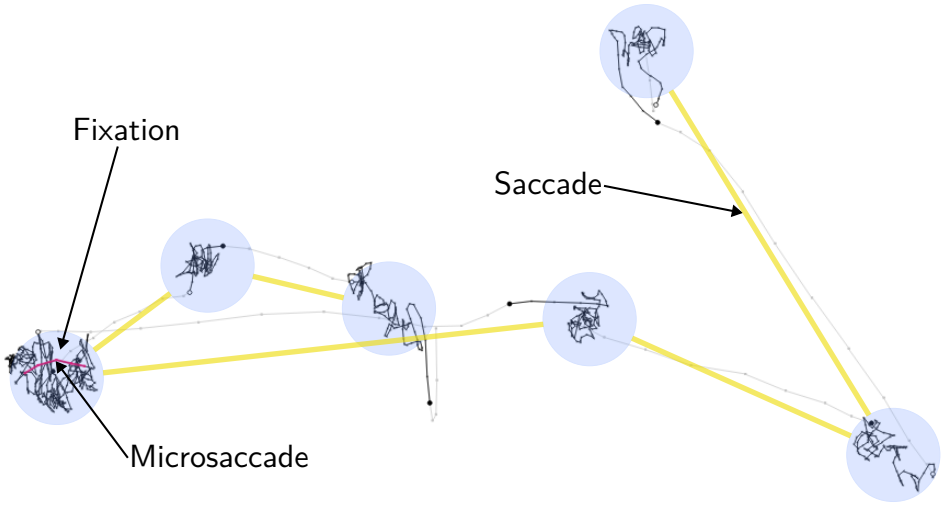


Figure 2.1 — Raw gaze positions are represented by dots connected by lines: gaze positions (together with their connections) that are labeled as fixations are black, as saccades gray, and as microsaccades pink. Each sequence of gaze positions that is marked as a fixation begins with a small white-filled circle and ends with a black circle. Additionally, an overlaid representation of a scanpath shows the averaged positions of fixations (blue) and saccades (yellow).

the exploration of the same data and may enjoy intuitive visualizations that are easy to understand and interact with. This may include finding patterns and gaining insights from a simpler system design with limited interaction.

2.4 Eye Tracking

One of the application domains considered in this thesis is eye tracking [158]. Eye tracking is used to record the eye movement of people looking at some stimulus. It is a typical example where real-world temporal data is collected. Visual analysis can here be used to gain deeper insights into eye movements and patterns related to cognitive aspects such as visual attention. Eye tracking is used in areas such as psychology, neuroscience, human-computer interaction, market research, and advertisement. The goals are, among others, to understand the visual attention and perception of humans and the improvement of the usability of interfaces.

Eye tracking data are often large datasets of time series consisting of gaze positions before feature extraction. Gaze positions are the locations on a screen or in a person's field of vision that someone is looking at. Figure 2.1 shows an example of the spatial locations of gaze positions. They are drawn as dots

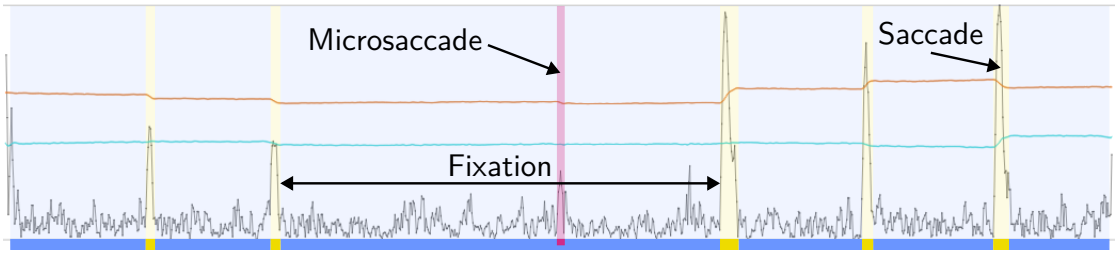


Figure 2.2 — A timeline illustrates the gaze positions in the x (orange) and y (turquoise) direction, as well as the velocity (gray). Additionally, the areas of fixations (blue), saccades (yellow), and microsaccades (pink) are marked. The same data as in Figure 2.1 is presented.

that are connected by lines to show the temporal connections. After extracting eye movements from the raw data, a scanpath illustrating basic eye movements (fixations and saccades) can be used to analyze these eye movements (circles and connecting lines in Figure 2.1). Figure 2.2 shows the same information that is visible in the scanpath (x and y directions of the gaze positions and eye movements) as a timeline.

Eye tracking data itself is relatively low-dimensional: for each time step, the coordinates where someone looked and possibly additional information such as pupil dilation. However, its analysis can be very challenging. Eye tracking experiments also include the stimulus, multiple participants, and different eye movements determined from the raw data (e.g., fixations, saccades, and microsaccades) that can be detected in various ways. Additionally, each eye tracking study has its own analysis or research goals that may require individual evaluation techniques in the analysis process.

This thesis uses eye tracking data as an example of temporal data. For their visual analysis, two approaches are presented (see Chapter 3). This section provides an overview of eye tracking properties, how data can be collected, a differentiation of eye movements, filters for their detection, and a survey of visualization and visual analysis approaches for gaze data.

2.4.1 Data Properties and Collection

Different types of eye tracking devices can be used to record the gaze positions of individuals for analysis purposes. The properties of the collected data vary, for example, in the type of stimulus being viewed, sampling rates, and the number of eyes being recorded. During recording, a series of positions (x and y values) with timestamps are captured, and sometimes additional properties such as pupil dilation.

The gaze is recorded with a specific sampling rate (frequency), depending on the eye tracker's hardware properties and analysis goals. The sampling rate defines the number of samples (gaze positions) recorded during one second; it is measured in Hertz (Hz). While for detecting some eye movements, a sampling rate of 30 Hz [108] can be sufficient, smaller eye movements require much higher recording frequencies for their detection. With the help of newer technology, it is even possible to record with frequencies up to 2000 Hz (occasionally even higher [131] when using specific hardware) and achieve results with very high accuracy. For all eye movements applies: the higher the sampling rate, the better and more precise their detection.

The gaze recording can be either monocularly or binocularly. In comparison, the first type captures only the movement of one eye, whereas the latter records both eyes. This influences the analysis since both eyes do not move in exactly the same way.

A crucial role in eye tracking is played by the accuracy and precision of recordings [158]. Accuracy is the offset between the recorded gaze position and the actual gaze position where someone is looking. Precision indicates the spread of repeated measurements of the same gaze position. These two properties are important for high-quality recordings. Therefore, eye tracking devices often have to be calibrated for individuals before recording to allow good recordings. These properties also highly depend on the tracking conditions. For example, when the subject's head is stabilized, and an experiment is performed in a controlled environment, the quality is usually better than when people can move freely with head-mounted devices while their gaze is being recorded. However, environmental factors like light sources and participant factors such as pupil size or glasses can also affect the recordings' precision and accuracy.

The movement of the eye is typically measured in visual degrees ($^\circ$) or minutes ($'$), with $1^\circ = 60'$. The *visual angle* represents the rotation of the eye that eye tracking systems measure. When working with stimuli presented on a monitor, eye tracking software usually outputs coordinates measured in pixels (px) on the screen rather than as a visual angle. Therefore, for the analysis, calculating the corresponding visual angles is often necessary. It is possible to calculate a value for pixels per degree (ppd) [px] that can be used to transform the gaze positions into visual angles. ppd can be determined from the distance to the screen (d [cm]), the width (or height) of the screen (w [cm]), and the horizontal (or vertical) resolution of the screen (r [px]):

$$ppd = \frac{r \cdot \pi}{360 \cdot \arctan(\frac{w}{2d})}$$

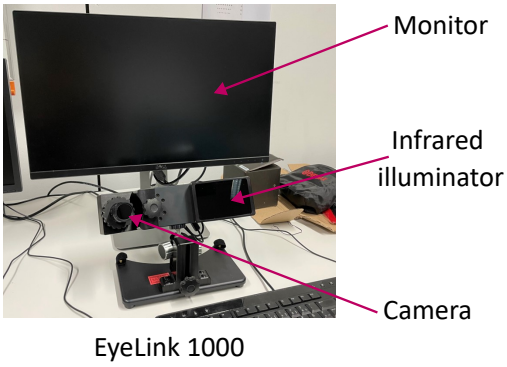


Figure 2.3 — A remote eye tracking setup consisting of a monitor, an infrared illuminator, and a camera to record the eye movements.

The visual angle α [°] can then be calculated for a pixel value v [px] (i.e., a distance measured in pixels) as follows:

$$\alpha = v / ppd$$

Most eye tracking devices can be classified into one of three categories: The first one utilizes attachments to the eye, such as special contact lenses with embedded mirrors or magnetic field sensors [108]. The movement of these attachments is measured. The next type measures electric potentials between electrodes placed around the eyes to detect changes in the eye position [108]. The final one operates without direct contact with the eye using optical tracking. In such a method, light (e.g., infrared) is reflected from the eye and detected by a camera or another optical sensor. The captured information is then analyzed to determine the eye rotation based on reflection variations. Video-based eye trackers are currently the most commonly used hardware solutions in eye tracking technology [108]. They belong to the third category previously mentioned. Such eye trackers are easily applicable due to their non-invasive nature (i.e., without contact with the eyes). These systems use cameras to capture videos of the eyes and analyze the movement of features like the pupil or corneal reflection [108].

This thesis uses only recordings from remote (or desk-mounted/stationary) video-based eye tracking devices. Typically, they consist of a monitor and a camera that records the reflection of the eyes illuminated by infrared light (see Figure 2.3). Using such a system, a person sits in front of the monitor, sometimes using a chin rest to stabilize the head and maintain a constant eye position during recording. In this thesis (Chapter 3), data from the Tobii T60XL is used for binocular recordings at a frequency of 60 Hz, from Tobii Pro Spectrum 1200 recording binocularly at 1200 Hz, and from SR Research EyeLink 1000 (Figure 2.3) recording binocularly at 500 Hz. Generally, this type of recording device allows very detailed and high-resolution recordings, and the recorded data is usually very accurate and precise. The higher the sampling rate, the

smaller eye movements can be detected and explored. Therefore, the first eye tracking device (Tobii T60XL) is better suited for an exploration of fixations and saccades, whereas the others also allow an investigation of very small eye movements such as microsaccades.

2.4.2 Eye Movements

Our eyes constantly move. When the gaze locations are recorded, coordinates of the gaze positions on the stimulus are captured. Afterward, the movement of our eyes can be aggregated into different eye movements for analysis (see Figure 2.1 and Figure 2.2 for some examples). Eye movements consist of voluntary and involuntary movements of the eyes [158]. These movements help acquire, fixate, and track visual stimuli. There are some basic types of voluntary eye movements used to explore the world around us: best known are fixations and saccades.

A *fixation* is the state when the eye remains more or less still over a period of time [158]: when our eyes fixate on one location, e.g., on an object. However, during this phase, the eye is never entirely still. Even when fixating, our eyes move slightly, but we do not notice it. Usually, fixations occupy a region of $0.5 - 2.0^{\circ}$, have a minimum duration of $50 - 250$ ms, and a maximum velocity of $10 - 50^{\circ}/\text{s}$ [158]. Three different types of micro-movements are distinct within fixations: tremors, drifts, and microsaccades [158]. They are known as intra-fixational eye movements, and all of them are involuntary.

The second voluntary eye movement is a *saccade* [158]. Saccades are very fast movements of both eyes simultaneously when they change the fixation position (e.g., from one object to another). Saccades are used to observe the world around us. They last about 30 to 80 ms, the velocity is typically in the range $30 - 100^{\circ}/\text{s}$, and they have an acceleration of $4000 - 8000^{\circ}/s^2$.

In contrast, the third type of voluntary eye movement, known as a *smooth pursuit* [158], is a smooth rather than a sudden movement. Smooth pursuits are much slower than saccades (a velocity less than $30 - 40^{\circ}/\text{s}$) and occur when following an object on a moving stimulus (e.g., following a bird across the sky). In this work, smooth pursuits are not considered. It is expected that the gaze data does not contain them. If they exist in the data, they might be misinterpreted as the filters used in this thesis do not recognize them.

As previously mentioned, there exist involuntary eye movements within fixations. Typically, we are not aware of any of these movements. They serve to correct the gaze to the fixation position. *Tremors* [158] are very fast and very small eye movements. They are the smallest eye movements, and their exact role is unclear [217]. *Drifts* [158] are slow movements that take the gaze

away from the center of a fixation. They play a role in sustaining precise visual fixation when microsaccades are absent [217]. These two eye movements occur between microsaccades [217]. *Microsaccades* [158] (sometimes referred to as *fixational saccades* [210]) are small and fast eye movements with a small amplitude, like tiny versions of normal saccades. They can be distinguished from regular saccades according to their amplitude and velocity. This type of eye movement plays a special role in this thesis. Prerequisites to detect them with an eye tracking system is a high frequency that should be more than 200 Hz [158]. However, most studies exploring microsaccades use devices with a frequency of 500 or even 1000 Hz (e.g., Engbert and Kliegl [111]) to capture them accurately. Their amplitude is not well defined: different information can be obtained from different published sources. However, most research uses a threshold of 1° for the amplitude (see Table 3.1 on page 60). Their contribution to visual perception is still not entirely understood. A few microsaccades serve the purpose of bringing the eye back to its original fixation location [158]. Others move the eye away from the intended fixation point, preventing the retinal image from fading [186]. It is also assumed that they are indicating covert attention. Previous studies have found that the location of attention of the viewer influences the rate and direction of microsaccades (e.g., Engbert and Kliegl [111], Engbert and Mergenthaler [113], Hafed and Clark [141], Hermens and Walker [149], Laubrock et al. [200], Pastukhov and Braun [247]). The presentation of an irrelevant stimulus or sound may also have an influence on microsaccades [186]. Properties of microsaccades that may be interesting during exploration include their direction (orientation), amplitude (angular distance), duration, velocity/speed, acceleration, curvature, position within fixations, and number of occurrences.

Besides these eye movements, some further movements very similar to microsaccades should be mentioned. Many saccades do not stop directly at the intended target, but the eye slightly fluctuates before coming to a stop. This post-saccadic movement is called *glissade* [158] (and sometimes *overshoot*). Glissades are quite common: between 20% and 40% of saccades end in glissades [158]. They are used to realign the eye before a steady fixation [336]. While glissades do not actually belong to fixations or saccades, in filtered data (using automatic eye movement detection algorithms), they are often located in the temporal areas classified as fixations, sometimes saccades. In the analysis of such data, the question arises as to whether these movements should be detected as microsaccades or not. The experiments in this thesis showed that the EyeLink gaze filter adds them to the fixations. Therefore, they are detected and characterized as microsaccades by many microsaccade detection algorithms applied to the temporal sequences of fixations. Additionally, the usage of the term glissade

varies across the literature. In various papers, these eye movements are handled differently (e.g., Bahill et al. [47], Deubel and Bridgeman [103], Weber and Daroff [336]), and it is not entirely clear what exactly belongs to this type of eye movement [158]. For example, Abadi and Gowen [30] do not differentiate between microsaccades and glissades. In this work, we regard as glissades all types of high-velocity over- and under-shoots directly succeeding saccades [158] (compare Section 3.1). *Square-wave jerks* [158] are another eye movement similar to microsaccades. Each square-wave jerk consists of a small saccade that moves the eye away from the visual target and a small saccade in the opposite direction back to the target. Therefore, they basically consist of pairs of small saccades (like two microsaccades) in opposite directions, parted by a short period of time (e.g., $200 - 400\,\text{ms}$ [120]). The physical appearance of small saccades, microsaccades, glissades, and the beginning and end of square-wave jerks are very similar; the same algorithms can determine all of these types of movements. Sometimes, the terms are even used interchangeably. Square-wave jerks are, for example, mentioned in the literature as a "pair of back-to-back opposing microsaccades" [141].

2.4.3 Eye Movement Filters

Many different approaches exist to detect eye movements. Typically, experienced eye movement researchers manually classify and label eye movements. This process is very time-consuming but usually very accurate. Some researchers only trust manually determined eye movements. Algorithms can help detect eye movements from the raw data containing gaze positions. These automated approaches consider properties such as velocity, acceleration, dispersion, and duration [158]. Methods based on properties require that the parameters for algorithms are well selected since, otherwise, the detected movements may be wrong. Modern methods also use ML to extract eye movements automatically [54]. After automated detection, eye movement researchers still have to verify that the detected eye movements are adequate, e.g., concerning the correct sequences of gaze positions used for labeling as some eye movement.

Usually, the software of eye tracking devices has built-in features for detecting basic eye movements and can immediately return the results for fixations and saccades, possibly also for blinks and other properties. Additional algorithms independent of these systems can also determine these and other eye movements.

There are two base types of algorithms that analyze the properties of the gaze to detect eye movements: Velocity- and acceleration-based approaches [158] (e.g., velocity-threshold fixation identification (I-VT) [276]) use velocity or acceleration

to detect eye movements; this is a method used, e.g., by Tobii and SR Research and that requires a minimum recording frequency of 200 Hz. This method can be used to detect both fixations and saccades. Dispersion- and duration-based algorithms [158] (e.g., dispersion-threshold identification (I-DT) [276]) use positional information and a form of clustering to detect fixations. They can especially be used for detecting fixations and also operate for recordings of a lower speed. For microsaccades, the algorithm by Engbert and Kliegl [111] is often used. This algorithm uses a velocity threshold and can be allocated to the first group of algorithms. Depending on the selected parameters, it can even be used to detect regular saccades. This algorithm is also described on page 62 in Section 3.1.3 for the microsaccade and saccade filter we use in VisME (Section 3.1).

A common approach for detecting fixations assumes that what is not a saccade is considered a fixation [158]. This is especially problematic when someone wants to explore microsaccades: glissades are then included in the temporal area of fixations. Automated analysis of these temporal intervals may then categorize glissades as microsaccades.

Since different algorithms exist for detecting different eye movements, and many of them have different parameters that influence the detection of eye movements, directly comparing the results between various studies and reproducing experiments is challenging. This topic is also explored in Section 3.1 for the case of microsaccades, where a method for reproducibility and verification of research results is presented.

2.4.4 Visual Analysis of Eye Tracking Data

Current software systems for eye tracking devices allow, besides the recording, the (visual) exploration of collected data (e.g., Tobii Pro Lab, EyeLink Data Viewer). However, in such a system, the analysis is often limited to basic visualizations. Specific analysis tasks frequently require unique visualization solutions.

The analysis of eye tracking data can be done with raw data (e.g., x and y positions), on filtered eye movements (e.g., fixations and saccades), or areas of interest (AOIs). In this thesis, only the first two methods are used.

There exist many different visualization techniques for eye tracking data [60]. Typically, timeline visualizations can be used to explore the temporal movement of the eye. Since eye tracking data consists of both temporal and spatial information, scanpaths (or gaze plots) [238, 239, 347] and attention maps (or heatmaps) [158, 294] are suitable to show detailed eye movements and summary visualizations of the gaze. Scanpaths show fixations and saccades as circles

connected by lines to visualize where a person looked. Overplotting might be an issue depending on the amount of data being visualized. Using animation and limiting the visualized elements to only a short time frame can circumvent this. Instead, attention maps summarize the gaze positions for a given interval or a whole trial to show where the highest attention of a person was. They highlight in the form of a color overlay where someone looked most. Timeline visualizations are used in Section 3.1, scanpaths in Section 3.1 and Section 3.2, and attention maps in Section 3.2.

Eye tracking data of one person is usually relatively low-dimensional (e.g., x and y coordinate, pupil dilation, duration of eye movements). Therefore, the previously described visualization approaches are well suited for exploring the data of one trial of one person in detail. Often, the analysis of the gaze of multiple people is required though, for which the comparative and aggregated analysis is usually more challenging. First, more data is available that has to be shown simultaneously for analysis and comparison. Second, it might be necessary to merge and summarize the data from different devices with possibly different hardware properties to present relevant information to the end-user (see Section 3.2). In general, beyond the eye tracking data, additional information might be included in the analysis. This could be the categorization of the trials into different tasks or additional temporal events (e.g., when something specific happened). Then, more advanced visualizations are required that also include these additional information.

2.5 Machine Learning

ML is a sub-field of artificial intelligence (AI) that is currently very popular and used in many different domains. Whereas AI has the general goal of imitating human intelligence, ML focuses on extracting information from data to solve specific tasks by identifying patterns. Application domains include NLP, computer vision (CV), and many more. In this thesis, both unsupervised and supervised learning methods are used. While unsupervised learning uses unlabeled data, supervised learning requires labeled data to train a model. For each category, many different approaches and algorithms exist with different properties and goals. While results created with ML approaches can be impressive, they are not always correct or as expected.

ML can be involved in the visual analysis process for sequential data in various ways. Data generated with ML approaches (e.g., internal states or prediction results) can be used as input data for data analysis, or ML approaches themselves can be used to analyze some data. In this thesis, approaches for both types are presented. Multiple projects use supervised learning techniques (particularly

DL) to generate data for analysis (see Chapter 4 and Section 5.1), while in others, unsupervised learning is used for the analysis process (see Chapter 5). Section 5.1 presents an approach that combines both methods: the input data originates from a classification model (supervised learning), and in the analysis, dimensionality reduction techniques (unsupervised learning) are used.

This section provides an overview of relevant ML background considering both unsupervised and supervised learning and some specific topics used in this thesis, such as dimensionality reduction techniques [320], NLP, NMT [297, 305], VQA [41], text classification, and different types of neural networks (NNs). Additionally, it provides a short overview of visual analysis used in the area of ML, especially targeting XAI to provide a better understanding of internal mechanisms of ML approaches.

2.5.1 Unsupervised Learning

In unsupervised learning, unlabeled data is used to learn patterns and structures in the original data. Common techniques include dimensionality reduction, clustering, and anomaly detection. In the following, only dimensionality reduction techniques are considered since only this unsupervised ML technique is used in the approaches presented in this thesis.

Dimensionality Reduction

Dimensionality reduction [320] (or dimension reduction) techniques are frequently used to reduce multi- or high-dimensional data to a lower-dimensional space while maintaining meaningful information. For generalization, we refer to this concept throughout this thesis as *projection*. Page 4 in Section 1 contains a brief explanation of why the use of this term is not always mathematically correct. Dimensionality reduction is often used as a preparation step before visualization when there are too many dimensions available to be visualized reasonably. Then, reducing high-dimensional data to two (or three) dimensions is beneficial for representing the data. An overview of this process is given in Figure 2.4. Corresponding visualizations can help interpret the underlying structure of the data, find patterns, and explore relationships between data points. It can be used, for example, to analyze similarities or differences between samples in a dataset. This thesis applies dimensionality reduction techniques in the preprocessing phase before visualization to project the multidimensional data to 2D (see Chapter 5). Then, each data instance is plotted as a dot. Besides this use case, a lower-dimensional representation of some data can also be used in various domains to save computational resources while working with only the most important information of the data. It is, for example, often used to

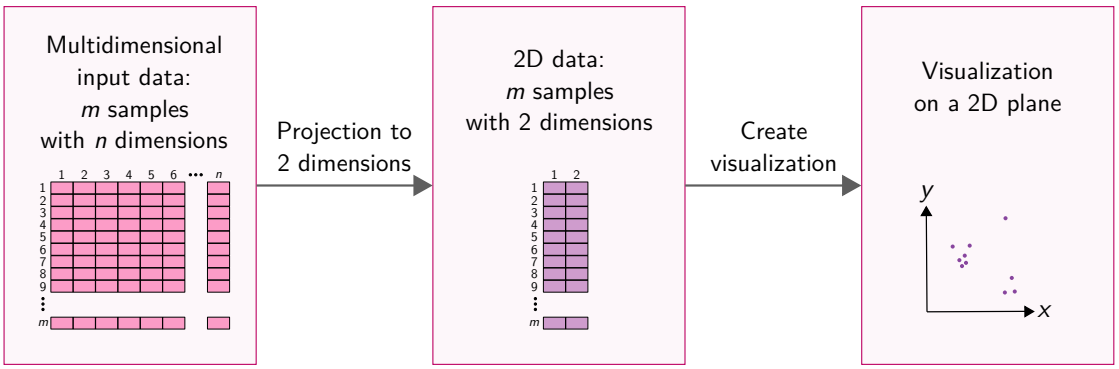


Figure 2.4 — Multidimensional input data consisting of m samples and n dimensions is used as input for a dimensionality reduction technique. After performing the projection, the resulting 2D data can be used to create a visualization on a 2D plane.

improve model performance as a data preparation step for supervised ML. It may, e.g., help reduce the training time [323].

While dimensionality reduction is a powerful technique, various factors and known problems should be considered when using this technique:

- By reducing the dimensionality, important information in the data may get lost. Therefore, essential structures and patterns in the data may no longer be available. After the projection, interpretability may be a challenge since some properties of the original data are no longer visible.
- The curse of dimensionality [107] is a challenge when dealing with high-dimensional data. With an increase in dimensions, the high-dimensional space gets sparser, and data samples are more spread out. Therefore, the distances between points get less meaningful. However, in the corresponding low-dimensional space, the meaning of distances becomes more important, leading to problems in accurately representing the structure of the underlying data.
- There is the problem of missing and false neighbors in the neighborhood of individual projected samples. Additionally, distances between samples might be inaccurate. This may result in artifacts distorting the visual representation and outliers that are not existing in the original data.
- Dimensionality reduction techniques differ in their linearity. Linear projection methods (such as PCA) use a linear data transformation. Therefore, distances between all projected data samples can be compared globally. This is often better suited during the interpretation of the projection. However, nonlinear properties of the original data may be missed. Here,

nonlinear dimensionality reduction methods (e.g., t-SNE) are better suited. The advantage of these techniques is that they can capture nonlinear relationships within the data. Such approaches try to maintain both local and global structures of the underlying data. However, results are often more challenging to interpret. The properties of the data (which are often not known before analysis) determine which type of method fits better for the analysis.

- Most dimensionality reduction methods use several hyperparameters that must be selected or adjusted before projection. The chosen values can significantly impact the quality of the approach. Many algorithms use a value from a random state as initialization for the random number generator. This number might have an effect on the outcome of the dimensionality reduction process. But there are other parameters, such as the chosen metric to calculate distances (e.g., the Euclidean distance is often used), or the perplexity and neighborhood value (determines the number of neighbors each data sample considers during the projection) for t-SNE and uniform manifold approximation and projection (UMAP), respectively. Such values have to be carefully chosen to generate meaningful results.
- Sometimes, applying a linear projection method (e.g., PCA) before using a nonlinear one may be useful. Especially if the number of dimensions is very high, the amount of data can be decreased by first reducing meaningless dimensions. This is a step recommended by Lee and Verleysen [203]. However, as mentioned before, this step might also lose important information.
- In the case of independent collection of different dimensions of the data, it may be helpful to scale the data before reducing it. A common method used for this purpose is standardization (e.g., z-score normalization) before applying the projection method. The result has a mean of zero and a standard deviation of one. This means that a positive value is larger than the mean, and a negative one is smaller. However, this technique assumes that the data has a normal distribution. If values have a low standard deviation, using this method could also negatively impact the quality of the projection (e.g., noise that gets too much importance) [203].

All of the previously mentioned features influence the final projection, its usefulness, and its interpretation. Choosing an adequate dimensionality reduction method is crucial, and each approach has advantages and disadvantages. For example, linear or nonlinear methods fit better depending on the properties of the input data.

There exist many methods to project data to a lower-dimensional space [320]. In the following, the focus is limited to the ones commonly used in the visual-

ization community. In this thesis, we mention the linear projection method PCA (which creates the same result as classical multidimensional scaling (MDS) [312] on Euclidean distances), and metric MDS as well as the nonlinear projection method t-SNE and UMAP. These methods are used in Chapter 5. In the following, the basic principles of these approaches are described.

Principal Component Analysis PCA [160] is a commonly used technique for dimensionality reduction in many fields. It is a linear projection method that performs a linear transformation of the data to a new coordinate system. The axes of this coordinate system, called the principal components, capture the largest variance in the data. Hence, this approach tries to find the most important features of a dataset to reduce the dimensionality. First, standardization is performed on the original data. Next, the covariance matrix is calculated, and eigenvalue decomposition is performed to find the eigenvectors and eigenvalues. Finally, the principal components are selected by ranking the eigenvectors, and the original data is projected on to these principal components. The principal components are linear combinations of the original features and are orthogonal to each other. Since this projection method is linear, it can be used to compare distances between samples globally and show how similar various data points are. In the resulting projection, samples that are closer together are likely to be more similar in the original high-dimensional space. However, some information may be lost in the process, especially variance along less important axes, and distances in the reduced space may not always perfectly reflect the true distances in the original space. Adjustable parameters of PCA implementations (e.g., the one of the scikit-learn [250] library) are, for example, the number of target dimensions (which is always two in our experiments since we want to use the projections for 2D visualizations) and a random state value (seed for the random number generator). Additionally, there are parameters for the solver (may influence, e.g., accuracy, speed, and memory usage), whitening (can be used to simplify the structure of the data), the number of iterations used during a computation step, and others.

Multidimensional Scaling MDS [64, 187, 312] has the goal of maintaining the pairwise similarities or dissimilarities between samples in a dataset when representing them in a low-dimensional space. As input, a distance matrix containing the distances between every pair of data samples is used. By minimizing a loss/cost function (e.g., a stress function), the new positions of the data samples are determined. The goal is that the distances in the lower-dimensional space are as close as possible to the distances in the original data. There are several types of MDS [64], including classical MDS, metric, and non-metric

MDS. Classical MDS [312] tries to keep the exact distances between the samples in the low-dimensional space using Euclidean distances for the distance matrix; it creates the same result as PCA [320] by using distance matrices as input instead of the exact values of the data instances. Metric MDS [187] is a superset of classical MDS representing a linear projection method that tries to maintain the dissimilarities between samples in the form of distances (often Euclidean distances are used, but other distance metrics can be used as well) by minimizing a stress function. In contrast, non-metric MDS [187] is a nonlinear approach that attempts to maintain the rank order of dissimilarities between data instances instead of the exact distances. The MDS method used in this thesis uses the Scaling by MAjorizing a COmplicated Function (SMACOF) [100] implementation. This nonlinear optimization algorithm uses an iterative optimization technique to minimize the cost function and find the best possible low-dimensional data representation. Each iteration step adjusts the distances between samples in the low-dimensional representation to approximate the original similarities or differences. Adjustable parameters of MDS implementations (e.g., the one of the scikit-learn [250] library) are, again, the number of target dimensions and a random state value. Additionally, there are parameters for the distance calculation between data instances (we use Euclidean distance), scaling (determines if metric or non-metric MDS should be used), and the number of iterations to run the algorithm.

t-distributed Stochastic Neighbor Embedding t-SNE [319] is a nonlinear dimensionality reduction technique mainly developed for visualizing high-dimensional data. The goal is to preserve the local structures of the data while also maintaining global structures. Therefore, it is beneficial for visualizing clusters or groups of similar instances. However, this also means that the global distances between samples cannot be used for comparison. For example, if two clusters exist, t-SNE may place them far apart, even if they are close in the original space. In this projection method, pairwise probabilities (representing similarities between points) between all data points in the high- and low-dimensional space are first calculated. Then, the positions of the individual points are adjusted by minimizing the Kullback-Leibler divergence between these distributions in an iterative way. Adjustable parameters of t-SNE implementations (e.g., the one of the scikit-learn [250] library) are, again, the number of target dimensions and a random state value. Additionally, there are parameters for perplexity (a value that determines the neighborhood size of data samples considered during the projection), the number of iterations used for the optimization, the distance metric (we use the Euclidean distance), and the learning rate (influences how fast t-SNE modifies the locations of data

samples related to the calculated gradients in each iteration). Such parameters highly influence the visualizations and their interpretation; projection results of t-SNE and the influence of parameter settings were explored by Wattenberg et al. [334].

Uniform Manifold Approximation and Projection UMAP [225] is also a nonlinear projection method, similar to t-SNE. It tries to maintain both local and global structures. This method is based on topology, where a combination of local and global neighborhoods are used to generate the lower-dimensional representation. After generating a neighborhood graph based on pairwise similarities in the high-dimensional space, repulsive and attractive forces are iteratively used to maintain local and global structures. Similar data points are pulled together, and dissimilar points are pushed away from each other. An advantage of UMAP is that it scales well for large datasets. Adjustable parameters of UMAP include the number of target dimensions, a random state value, a distance metric, and a value for the neighborhood size. Additional parameters are, for example, a value that defines how tightly points are packed together (the minimum distance points are allowed to be apart). This parameter can help if someone is interested in either clustering or a broader topological structure.

These dimensionality reduction techniques were initially used by the visualization community to map multidimensional data to 2D or 3D for visualization. This was usually done by drawing data samples on static images or in interactive 3D grids. Recently, dimensionality reduction methods were also more often applied to sequential data (see Table 5.1 on page 171). Here, the visualizations of the dimension reduced data can consist of a point cloud and additional lines that connect subsequent points. While there are already interpretation problems for static data, they appear more severe for sequential data, especially since additional visual elements influence the interpretation of the data. More details on this issue follow in Section 5.2.

2.5.2 Supervised Learning

Supervised learning uses labeled data to train a model. This means that the training data consists of pairs of input data (e.g., an image or text sequence) and output values (e.g., a category). Then, the model is trained with such exemplary data. The goal of the model is to generate correct predictions for new data samples: given new input data, an output value is predicted. Common algorithms include support vector machines (SVMs) [93], regression [95], decision trees [259], random forests [66], k-nearest neighbors, and NNs [31]. While there

exist many different algorithms [241], this thesis only uses NNs. Goals that can be solved with such approaches are usually classification and regression tasks. Classification models predict, for an instance, one of several discrete categories or classes. In contrast, regression models predict a numerical value rather than a category.

In NNs, different aspects related to sequential components can be found. First, the input data can be sequential (e.g., text input). Next, the training is a temporal process. Finally, internal processes can be performed sequentially (e.g., processing the text input), and the output may be a sequence. All of the projects presented in this thesis use textual data as input that is processed in sequential order. In these approaches, complex internal behaviors of ML have to be considered, and essential information is presented to the users for analysis, exploration, and debugging. Visualizing text sequences and the models' internal states must often be combined. In addition, internal information, possibly without a sequential component, has to be extracted to be visualized for analysis in combination with the sequential information.

This thesis uses only ML models in the context of NLP, namely, NMT, VQA, and text classification tasks. Of particular interest are DL approaches, especially RNN (long short-term memory (LSTM) [156]) and Transformer [322] models. They are all specialized in processing sequential data and can be used as sequence-to-sequence (Seq2Seq) [86] models. Another model used in this thesis that is not specialized for sequential data is a graph neural network (GNN) [341] and will also be briefly described in the following. A big challenge of some supervised ML (especially DL) approaches is their interpretability. ML approaches are frequently experienced as black boxes. They can provide impressive but sometimes wrong results. Often, it is unclear to users how the predictions were made. Therefore, one goal of this thesis is to provide visual analysis solutions to better understand internal mechanisms and prediction results of such approaches.

Natural Language Processing

NLP is closely related to AI. One type of approach involves processing natural language (e.g., text and speech) with the help of ML techniques. Other AI-based approaches, such as Symbolic AI, are also used in this domain but are not further considered in this work. Common tasks in NLP include machine translation of a text (e.g., NMT), question answering (e.g., in the context of VQA), text classification, text summarization, text generation, and speech recognition. In this thesis, approaches for NMT, VQA, and text classification are explored.

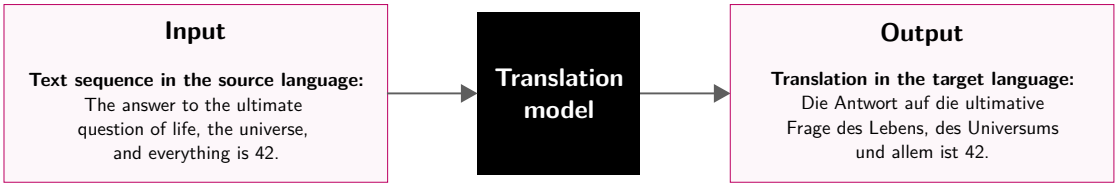


Figure 2.5 — To translate a text sequence from a given language to another language, a translation model automatically generates the translation in the target language.

Often, in NLP, NNs (e.g., DL models) are used to process text. Therefore, a large amount of annotated data that contains text sequences is used during training. While NLP also uses further techniques, such as statistical or rule-based approaches to enhance language understanding, this thesis only focuses on the ML parts. Similar to other ML areas, the problems of understandability, interpretability, and debugging are challenging since the models are considered black boxes.

Neural Machine Translation As it becomes increasingly important to communicate in different languages, and since information should be available to many people from different countries, many texts must be translated. Human translation is a very time-consuming process. Computers can help speed up this process. In the context of DL, NMT [297, 305] is a powerful option to realize this. NMT automatically translates a sequence of words from one language into a sequence of words in another language using NNs (see Figure 2.5). An input sequence consists of source tokens from a source vocabulary and an output sequence of target tokens from a target vocabulary. A huge amount of training data is required consisting of input sentences and their translations to generate a good prediction model and create appropriate translations.

Different approaches exist to achieve this goal [305, 344]. The DL model must be able to process and memorize the content of long sequences. Usually, Seq2Seq models are based on models designed to deal with such sequential data. Therefore, LSTMs and the Transformer architecture are well-suited (see below).

One of the first NN architectures for machine translation consists of two RNNs with LSTM units [86]. There are extensions to this basic architecture to generate better predictions. This includes, for example, an attention mechanism [46] that is added to the LSTM architecture. Such an attention mechanism allows the focus of the model on different parts of the input sequence. Also refer to page 46 for more details on the internal mechanisms of a Seq2Seq model.

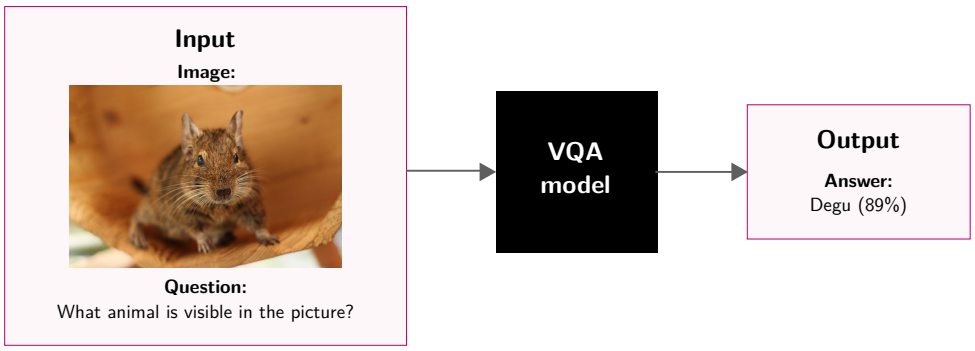


Figure 2.6 — A VQA system takes an image and a question as input and predicts the answer.

With the introduction of DL methods, the translation quality of machine translation models has improved considerably in the last few years. However, there are still difficulties that need to be addressed. Common problems of NMT models are, for instance, over- and under-translation [314] when words are translated repeatedly or not at all. Handling rare words [183], which might be available in specific documents and long sentences, is also an issue. Domain adaption [183] is another challenge. Especially documents from specific domains such as medicine, law, or science require high-quality translations [90]. As many NMT models are trained on general datasets, their translation performance is worse for domain-specific texts. In the approach presented in Section 4.1, we address these issues and present a visual analysis solution where it is possible to explore translations and improve the NMT model.

Visual Question Answering VQA [41] combines concepts from both NLP and CV. Given an image and a natural language question, a VQA system automatically generates an answer for the question (see Figure 2.6). This means, a user can ask an arbitrary textual question related to an image, and a model provides an answer. Typically, VQA systems rely on DL approaches [342]: the image and the question are separately processed to extract features that are then used to generate the final prediction for the answer. Often, attention mechanisms are used to focus on the most important areas of images and the questions. VQA is usually treated as a classification problem, i.e., the model predicts an answer from a predefined set of answers [169]. The answer with the highest probability or confidence score is selected as the final prediction. For training, a large amount of images, questions, and corresponding answers is required. A novel approach to enhance the understanding of the visual content is based on graph-based VQA [97]. Here, images are not directly fed into the model. Instead, a scene graph [81] representation of each image is required.

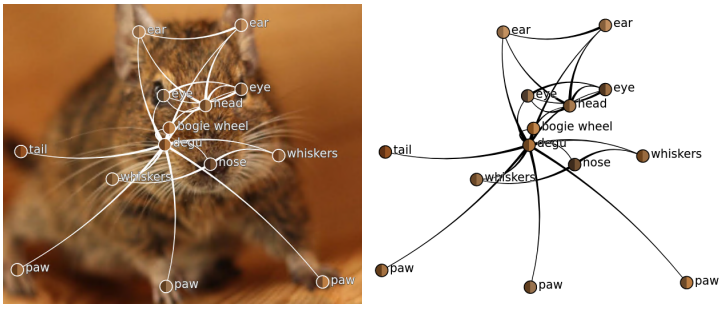


Figure 2.7 — Example of a scene graph overlaid on an image (left) and without image (right).

Scene graphs (see Figure 2.7) are directed graphs consisting of nodes and edges that describe the content of images. The nodes represent the objects in an image with their attributes, and the edges describe relationships among neighboring objects. To process such scene graphs, GNNs (see page 48) can be leveraged, which are special types of NNs specifically designed to process graphs. In Section 4.2, we present an approach to analyze a graph-based VQA system [97]. In this approach, we employ a graph attention network (GAT) [70] (a specific type of GNN), where each edge is assigned an attention value that can be used in the analysis to better understand the prediction mechanism.

Text Classification Classification is the process of categorizing data into different predefined classes or categories. There are different ML approaches available to achieve this (e.g., NNs, decision trees, SVMs). Depending on the input data and expected performance, one of these methods might fit better than the others. ClaVis [6] is a system that can help compare different classifiers regarding performance and other properties. For text classification, the input is a text sequence, and the output is a category (see Figure 2.8). A specific type of text classification is sentiment analysis. Here, the sentiment or opinion of a text is detected; the categories may include, e.g., the terms *positive* and *negative* reflecting the opinion of someone writing a text sequence. Before entering the text sequence into an ML model, it has to be transformed into a vector representation that is used as input. In DL, common prediction approaches are based on RNNs or convolutional neural networks (CNNs) to process the text sequences and obtain an activation vector. The final prediction is then generated with a classification layer, e.g., a fully connected layer, to map the activation vector to the possible categories. As activation functions, sigmoid for binary classification and softmax for multi-class classification can be used. The output of this layer is a probability distribution over all classes. The final prediction is the class with the highest probability. During training, text sequences and the corresponding classes are fed into a model. In Section 5.1, we present a visual analysis approach for text classification based on LSTMs. Here, users are

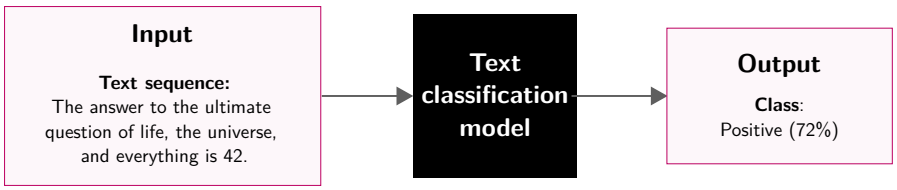


Figure 2.8 — For text classification, a text sequence is passed to a text classification model, which outputs a class.

supported in analyzing the hidden states of an LSTM used to process the input sequences.

Artificial Neural Networks

Artificial neural networks (ANNs) [31], or more commonly abbreviated as NNs, are an important concept of ML. NNs are inspired by the structure and functionality of biological NNs available in the human brain. They consist of nodes (called artificial neurons) connected by edges. Each neuron and edge has learnable parameters: each node has the bias parameter, and each edge has a weight. The nodes are usually aggregated into multiple layers, and each node has inputs and outputs. The information flows through such a network from the input layer (i.e., the input neurons) to the output layer (i.e., the output neurons). See Figure 2.9 for an example. The layers between the input and output layers are called hidden layers. Input values are passed to neurons that apply an activation function to generate an output and pass the result to neurons of the next layer. An example for a hidden layer is a fully connected layer (also known as linear layer or dense layer). Here, each neuron of the layer is connected to every neuron of the previous layer. An activation function uses the weighted sum of inputs of the neurons plus the bias of the actual neuron to calculate the output (also called activation vector). Commonly used activation functions include sigmoid, softmax, rectified linear unit (ReLU), and tanh. The choice for the activation function depends on the problem the network needs to solve. Sigmoid and softmax are usually used in the output layer: sigmoid is commonly used for binary classification tasks, and softmax for multi-class classification since it provides a probability distribution over different classes. ReLU and tanh are more commonly used in hidden layers. Here, ReLU is often found in CNNs, while tanh is applied in RNNs and LSTMs.

The parameters of a network (weights and biases) are learned during the training process of an NN, where training data (consisting of input and output values) is fed into the network. These parameters are adjusted using a loss function and an optimization algorithm to minimize the error between predicted

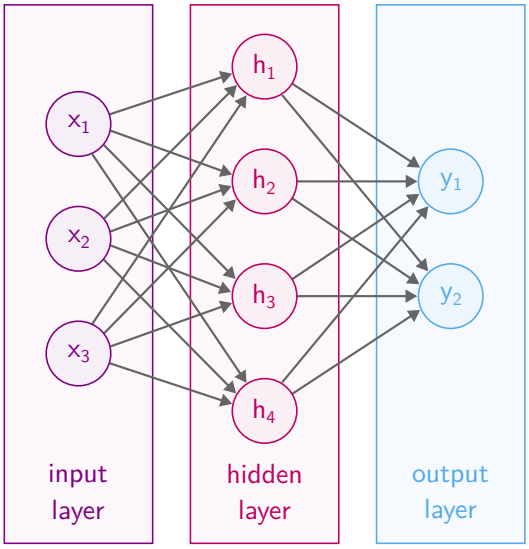


Figure 2.9 — Example of a simple NN (FNN) consisting of an input layer, an output layer, and a hidden layer. Each circle represents one neuron, and each arrow depicts a connection between two neurons.

and expected outputs. This means they are adapted in such a way that when feeding samples of the training data into the network, the correct prediction should be performed. A loss function (e.g., mean square error for regression or cross entropy for classification tasks) compares the difference between the prediction and the correct value. Backpropagation is a commonly used method to minimize error value by propagating the error backward through the network to compute gradients of the loss function concerning each parameter. Then, an optimization algorithm, such as gradient descent, is used to update the parameters of the network to minimize the results of the loss function such that the network learns to perform better predictions. To avoid overfitting, a previously specified percentage of random neurons in each layer can be ignored during training (dropout). Typically, the training involves an iterative process consisting of multiple passes or epochs. In each epoch, the entire training dataset is processed. This is often done using multiple iterations, where a number of training samples (specified through a batch size) are used to update the model. Multiple iterations are needed to process all training data within one epoch. With the help of batches, the model weights are updated after each iteration, not after each training sample or after processing the whole dataset. Usually, a huge amount of training data is required to generate good models.

There are many different architectures for NNs. An example are feedforward neural networks (FNNs) (see below). Deep neural networks (DNNs) are NNs containing multiple hidden layers. These layers are located between the input layer and the output layer. It is not well-defined how many such layers are required to count an NN as DNN. DNNs are an architecture belonging to the concept of DL. Typical examples of DNNs are RNNs (e.g., LSTMs), CNNs,

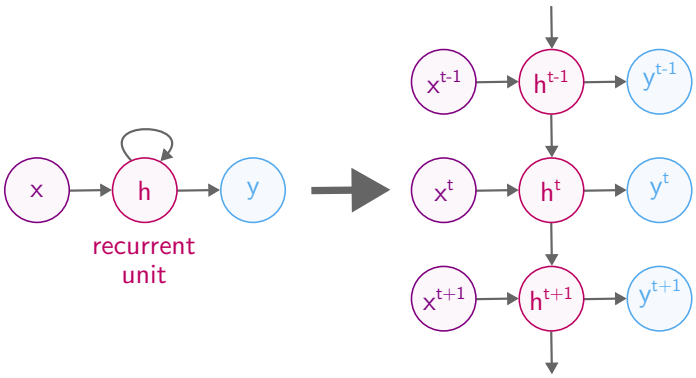


Figure 2.10 — Example of an RNN. It is visible that the current hidden state is computed from the previous hidden state and the current input.

GATs, and Transformers. Other architectures exist, and many combinations and extensions thereof. In the following, the ones of importance to this thesis are presented.

Feedforward Neural Network An FNN is a simple form of NN. The information is passed to an input layer and may flow over hidden layers to an output layer (see Figure 2.9). The main characteristic of this concept is that information flows only in one direction. There are no cycles that may, for example, occur in RNNs.

A multilayer perceptron (MLP) is one type of FNN. It is a fully connected NN where each neuron of a layer is connected to each neuron of the next layer. CNNs are another type of FNN, often used in CV, where the input is a matrix. They contain one or more convolutional layers followed by other layers, such as pooling or fully connected ones. The convolutional layers use convolution operations on the input data with convolution kernels to extract features; the results are feature maps (or activation maps), which are passed to the next layer. The pooling layer is used to downsample the map by removing unimportant information.

While FNNs are very common, in the context of this thesis, we do not investigate them directly. They are not well suited for processing sequential data. However, they are used within larger NN architectures (e.g., the Transformer).

Recurrent Neural Network RNNs are NNs developed to process sequential data. They are an often used technique in NLP where text sequences are processed. This is enabled through the internal memory of the model at a time step in the form of a vector (a *hidden state*); it contains information about the already processed sequence. The architecture of an RNN consists of connected units that use an input and the previous hidden state to produce an output and a new hidden state (see Figure 2.10). The dimensionality of a hidden

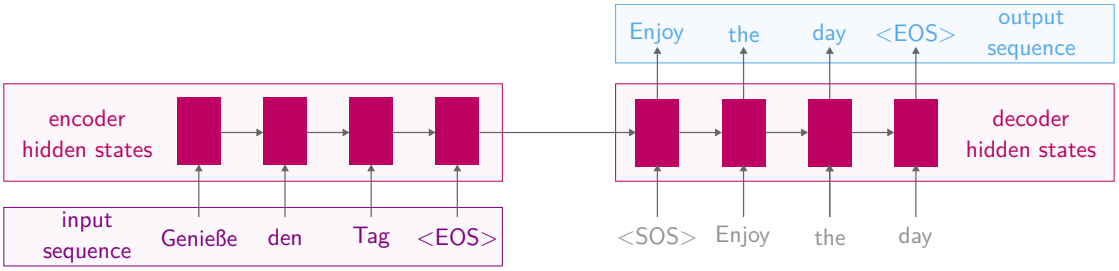


Figure 2.11 — Example to illustrate a basic Seq2Seq model according to Cho et al. [86] consisting of an encoder and a decoder to translate an input sequence into another language. The encoder reads each term of the input sequence. At each step, the previous hidden state and the next term of the sequence are used to determine the next hidden state. The decoder starts using the last hidden state of the encoder together with the start-of-sequence (SOS) token to generate the first term of the output sequence. It continues generating tokens using the token and hidden state determined previously until the end-of-sequence (EOS) token is reached.

state is defined by the number of units or neurons in the hidden layer of an RNN; they can be individually defined. Input can be, for example, the individual words of a text sequence. The problem of RNNs is that long-term dependencies cannot be well captured due to the vanishing gradient problem. This problem occurs during the training of an NN when the gradients get very small during backpropagation through multiple layers. Therefore, there are multiple improvements for RNNs.

LSTMs [156] are an extension of traditional RNNs that can handle long sequences. They use a gating mechanism to handle which information should be retained or forgotten. There are three gates: the input gate, the forget gate, and the output gate, used to forward and forget information. Often, LSTMs are combined with an attention mechanism [46] to improve their performance. LSTMs are used in Section 4.1 in the context of NMT and Section 5.1 for text classification.

In the context of NMT, a basic Seq2Seq model [86] consists of two RNNs (e.g., with LSTM units): an encoder and a decoder (see Figure 2.11). While the encoder transforms the source sequence into a fixed-length representation (a context vector), the decoder generates a target sequence based on this vector. The length of the input and output sequence do not have to be the same. An extension of such a basic model uses an attention mechanism [46], which allows the model to focus on different parts of the input sequence during the prediction of the next element of the target sequence by providing the decoder access to the hidden states of the encoder. An attention mechanism enables different

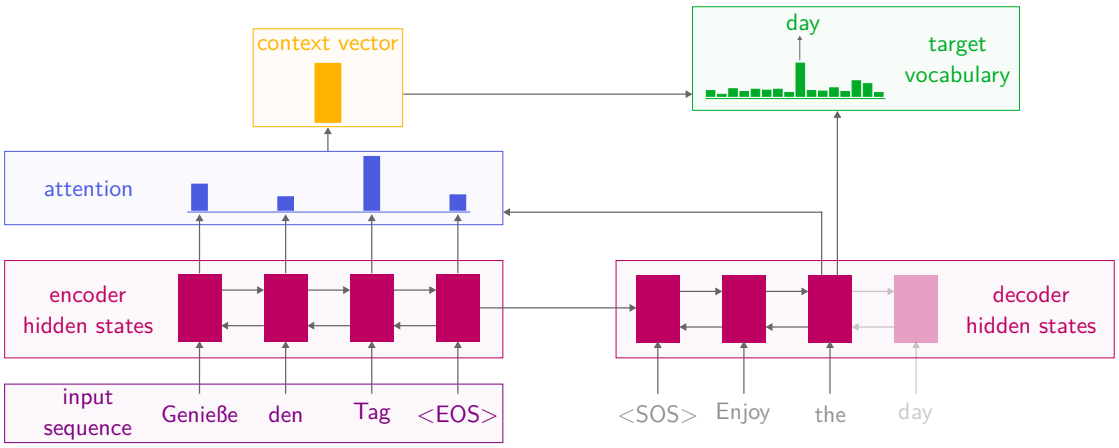


Figure 2.12 — Example to illustrate one step of a Seq2Seq model consisting of a bidirectional encoder and decoder architecture, and which uses an attention mechanism to translate an input sequence into another language. The encoder generates hidden states for each token (by concatenating the forward hidden state (produced by a forward RNN reading the sequence from left to right) and backward hidden state (produced by a backward RNN reading the sequence from right to left)). The encoder hidden states and the current decoder hidden state (for the term "the") are used to determine the context vector, which is used to predict the next word ("day"). Illustration modified from See et al. [282].

parts of the input sequence to be assigned different weights according to their importance. Then, a weighted sum of the hidden states can be computed such that the model focuses more on relevant information. Figure 2.12 shows an example for one step during the prediction. Here, attention weights are determined from the current decoder and encoder hidden states. Then, a context vector for the current step is determined as a weighted sum of the encoder's hidden states using the attention weights. This context vector and the current decoder state are used to generate the output. The final prediction is obtained as a probability distribution over the target vocabulary, based on the context vector and current decoder state. Attention weights represent a soft alignment between source and target sequences. The weight distribution over the source sequence shows how the source words contributed to the prediction for each translated word.

Transformer The Transformer architecture [322] is a very popular ML technique based on NNs. Results created with this technique are usually impressive. It is used in many different domains and has led to the development of several further methods (e.g., Bidirectional Encoder Representations from Transformers (BERT) [104], Generative Pre-trained Transformer (GPT) [261]). Similar to

LSTMs, the Transformer was developed to process long sequences. Here, the attention mechanism plays a very important role. Compared to the previously mentioned attention mechanism of LSTMs, the Transformer uses a self-attention mechanism which is used extensively throughout the model to efficiently capture long-range dependencies in sequential data. The Transformer is especially well suited for machine translation and other areas where Seq2Seq operations have to be performed. Key elements of such an architecture are an encoder and a decoder. The encoder transforms the input sequence into a context vector, and the decoder creates an output based on this context vector. Both the encoder and decoder consist of multiple layers. The layers of the encoder consist of a self-attention layer (determines attention scores for each element of the input) and a fully connected FNN. The layers of the decoder consist of a self-attention layer (determines attention scores for already generated elements in the output), a fully connected FNN, and an encoder-decoder attention layer (allows a focus on important parts of the input sequence encoded by the encoder). Multi-head attention allows the model to focus on the input sequence simultaneously multiple times for the computation of attention scores; the results are combined and transformed into a final output. It is used for both self-attention and encoder-decoder attention. The Transformer architecture is used in Section 4.1.

Graph Neural Network A GNN [341] is a type of NN designed for graph data. As input, a directed graph is used, consisting of a set of nodes and a set of edges connecting the nodes. The nodes represent objects, and the edges are the relations between them. In this thesis, we use GATs [70], a specific type of GNNs. A key feature of this architecture is the use of attention mechanisms to capture important relationships within a graph. In this work, graph classification is performed by the model.

The key concept of GNNs is message passing, where information between neighboring nodes is exchanged to update their node representations (i.e., feature vectors or node embeddings) based on learned edge attention weights. Here, multiple layers are used to perform graph convolutions; in each layer, the node representation of each node is updated. After multiple convolutions, the node embeddings can be used to make the final prediction. Compared to other GNN architectures, a characteristic of GATs is their ability to weigh incoming edges differently. Information is passed along graph edges multiple times, namely in each GNN layer. In each layer, it is possible to record the weight shares of all edges. These values allow users to get an impression of which edges contributed how strongly to the updated value of any given node in every step. Additionally, a graph gating mechanism is used by GNNs to aggregate the

node embeddings into a single vector representation for the prediction. Here, the global node attention values (graph gate weights) are computed between a global language representation and the node embeddings. Nodes with higher attention values contribute more to the final result, while nodes with lower values have less influence. We use these information in our approach presented in Section 4.2.

2.5.3 Visual Analysis of Machine Learning

ML techniques can create impressive prediction results, but results are still sometimes incorrect. Visualization and especially visual analysis are helpful approaches to increase the transparency and interpretability of ML models, facilitate debugging of predictions, and even improve model quality through user input. This concept is often referred to as the current research topic visualization for machine learning (Vis4ML) [330], a subset of XAI [50]; it can visually help understand why certain predictions were made. While XAI helps to better understand and improve the transparency of AI model outputs through various techniques, Vis4ML uses visualizations to gain insights into prediction results and the inner workings of a model. By helping users of ML models better understand and interpret model decisions and their relationship to the input data, trust in these models can be increased. Figure 2.13 illustrates the connectivity between the prediction mechanism of a trained ML model and visualizations that can be created from the input, output, and the internal states of the ML model with examples presented in this thesis.

Different strategies can be chosen to analyze ML models [130]. One approach is to explore a model's internal states while performing a prediction. This helps see how these states change and contribute to the final prediction. Another method is to explore the actual prediction results that can provide insight into the prediction and provide a quality assessment of the performance of the model. Other approaches, such as exploring the training process (e.g., Nguyen [234]), are also possible but are not further considered in this thesis. The strategies can be applied to a variety of data types. In the context of this thesis, we focus on analyzing DL approaches and chose examples from NLP.

ML models are often seen as black boxes and are hard to understand and interpret. Their internals might be complex, consisting of many calculation steps, and may not be easily interpretable by humans. Therefore, one of the main goals in this area is to open the black box of ML, making ML models more transparent and controllable. The visualization and interpretation of internal states of NNs is currently a popular research area. Visual analysis methods were developed and evaluated to achieve this, allowing users to better understand, improve,

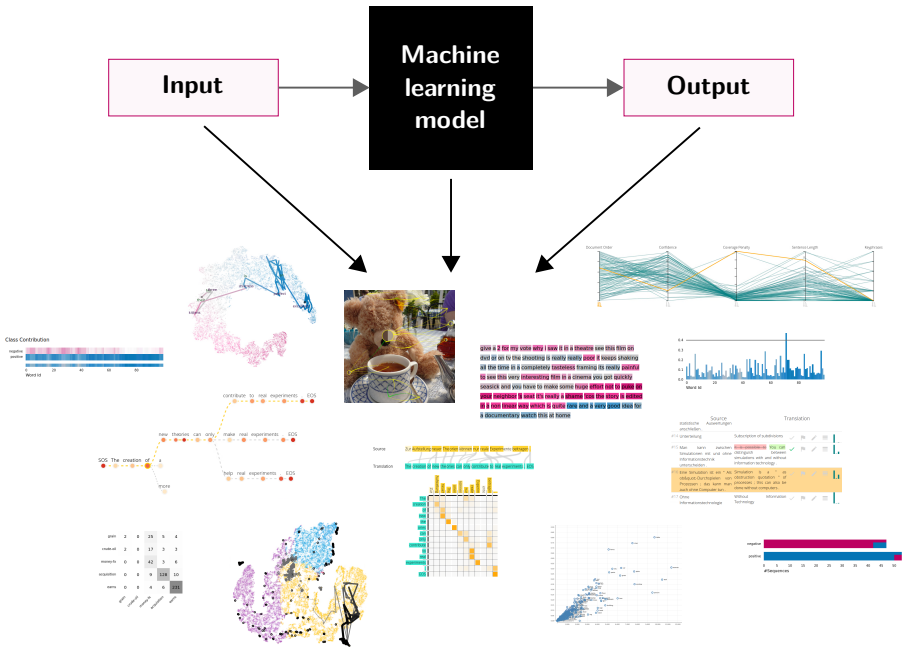


Figure 2.13 — Input, output, and internal states of a machine learning model can be used to create visualizations in the context of Vis4ML; the visualizations were created with the approaches presented in this thesis.

and control ML models, especially DL models. There are many survey papers showing the current interest in this area (e.g., Alicioglu and Sun [36], Beauxis-Aussalet et al. [53], Choo and Liu [89], Garcia et al. [130], Hohman et al. [157], La Rosa et al. [197], Liu et al. [209], Yuan et al. [351]). Most of these publications present an overview of existing work and a new taxonomy to classify existing work or report on present or future research directions. Beyond these works, numerous individual solutions exist for different problems to deal with ML aspects such as transparency, interpretability, and trust. Approaches developed in this context aim to show the features learned and to make the learning and decision process more transparent and controllable. This is achieved by giving humans who have the goal to understand the model decisions and improve the final performance access to internal information of ML models.

In this thesis, the idea of using ML approaches (e.g., unsupervised ML) to create visualizations to better understand other ML approaches (e.g., supervised ML) is also pursued. This means that ML is applied to generate visualization of ML aspects. By extending the concept of Vis4ML to include the additional use of ML for creating visualizations, we introduce the term machine learning for visualization for machine learning (ML4Vis4ML). Such an approach is used in Section 5.1. Here, a supervised ML technique (text classification based on

LSTMs) was used, and visualizations from internal states of the classification were created by performing unsupervised ML (dimensionality reduction) for exploration and debugging purposes. It would even be possible to extend this approach by applying supervised ML on results from supervised ML to create visualizations. This might bring unexpected insights into various ML models.

Chapter 4 and Section 5.1 present different visual analysis solutions for the exploration of internal states and the final prediction results of NMT, VQA, and text classification. The work by Chatzimparmpas et al. [84] presents a state-of-the-art report on interactive visualization approaches that enhance trust in ML. The DL-based approaches concerning NMT and text classification presented in this thesis [5, 15, 19] (Sections 4.1 and 5.1) are also listed in the online version[1] of their survey. Additionally, the approaches presented in this paper include temporal visualizations. The work about time series projections generated with dimensionality reduction [22] (see Section 5.2) is listed in the online version of the TimeViz Browser 2.0[2] [310].)

2.6 Determining Requirements

Throughout this thesis, we outline specific requirements for each visual analysis approach, serving as the foundation for our implementations. These requirements were usually based on multiple sources. First, they were built from challenges identified within the specific research areas and analysis goals. Then, characteristics were analyzed to formulate requirements for addressing them during the development and evaluation of the visual analysis approaches. Second, we also based the requirements on the concept of design as domain experts [127] with long-term, in-depth experience: We designed our visualization techniques in the context of large-scale and long-term collaborative projects (SimTech[3] and SFB-TRR 161[4]). These projects deal with many topics also explored in this thesis: techniques and applications for eye tracking, XAI, including supervised ML (e.g., NMT and VQA) and unsupervised ML (e.g., dimensionality reduction). For most analysis systems presented in this thesis, a collaboration with experts in areas such as eye tracking (Section 3.1 and Section 3.2), DL and NMT (Section 4.1 and Section 4.2) was conducted. The

[1] TrustMLVis Browser, https://trustmlvis.lnu.se/

[2] The TimeViz Browser 2.0, https://browser.timeviz.net/

[3] Cluster of Excellence "Data-Integrated Simulation Science (SimTech)", www.simtech.uni-stuttgart.de

[4] DFG Transregional Collaborative Research Center SFB-TRR 161 "Quantitative Methods for Visual Computing", www.sfbtrr161.de

development of these systems involved an iterative process with multiple meetings with these domain experts to collect feedback and improve the analysis systems.

2.7 Evaluation Methods

In this thesis, different types of evaluations are performed to assess the usefulness of the presented visual analysis systems. Empirical studies are a key method for new visualization approaches or visual analysis systems to be evaluated [51, 77, 167, 198]. It is possible to show the usefulness through expert studies, e.g., using a think-aloud protocol analysis [117] or other types of usability studies [124]. Instead, automated computer-based analysis using some metrics for evaluation or a demonstration of use cases can be employed to evaluate an approach. Often, it is useful to use a combination of different evaluation methods [77] to help better understand the efficiency of a visualization approach from various angles. The different types of evaluation methods can be categorized as *qualitative* and *quantitative* evaluation [77].

Qualitative Evaluation

In a qualitative evaluation, non-numerical data is collected to gain insights into subjective experiences, perceptions, and behaviors. Such data can be collected as observations, in interviews, or in questionnaires.

One approach are user studies [124]. This method is based on observing, for example, visualization or domain experts using a system and collecting their feedback. Here, a common technique is a think-aloud protocol analysis [117], where experts or other users can use and explore the system. Users verbally express their thoughts while engaging with the system. They report what they do, if they have problems, or what they like. Additionally, feedback, comments, and suggestions can be collected through an interview or a feedback form after using the system. We used this technique, for example, in Section 3.1.

In questionnaires and expert interviews, users can answer specific questions to evaluate their answers later. The standardized questionnaire on usability, System Usability Scale (SUS) [71], is an often applied and adopted method. It consists of 10 questions about the usability of software. The user feedback is collected on a Likert scale [207] from 1 (strongly disagree) to 5 (strongly agree). Different systems can be compared with this questionnaire. Additionally, software or approach-specific questions can be used or added to other questionnaires to collect more specific responses concerning the features and functionalities of the software being evaluated. While this type of evaluation can

create quantitative results, the interpretation may usually rely on the subjective opinions of participants, and therefore, results might be analyzed qualitatively. In such questionnaires, some questions may ask for objective data (such as age), providing quantitative information. Besides numerical answer options, users can also answer verbally in free text forms of a questionnaire or in interviews to collect general feedback about an approach.

Case studies [167] and a demonstration of use cases can also be employed to show the capabilities of a system. A case study offers an in-depth analysis of how an approach is used to address a real-world problem, often involving a collaboration with domain experts. It provides detailed insights into the approach, frequently involving an iterative process to improve the application. In the literature, case studies may also report on how researchers apply an approach [167]. However, in these cases, according to Isenberg et al. [167], a more appropriate term to describe this might be usage scenarios. In this thesis, we show in several sections examples of the capabilities of our systems and how they can be used to achieve some goals; we refer to these types of evaluations as demonstrations of use cases or examples (e.g., in Section 5.2.4).

Quantitative Evaluation

In a quantitative evaluation, measurable numerical data is collected. A statistical analysis can then be used to draw conclusions and make decisions based on the quantitative measures.

For example, quantitative data can be collected by measuring metrics of participants in regard to their performance when completing a task [198]; this includes response times or error rates when using a system. An overview of how well participants could work with different parts of a software system (e.g., when completing different tasks) and differences between participants can be shown through analysis of such performance values. We use this technique in Section 4.2.

Another method in quantitative evaluation is automated computer-based analysis to evaluate the performance of an approach according to some metrics. User behavior can be imitated or simulated to test if some software has the expected result under different conditions. Here, a large amount of data can be generated and processed to gain insights into the performance and reliability of an approach. We do this for example in Section 4.1.4. Instead, it is also possible to automatically evaluate the relevance of the information presented in the visualizations. On page 140 in Section 4.2.3, an example of such an evaluation is shown.

There are further methods of quantitative evaluation that were not used in this thesis. An example is the use of eye tracking to record and analyze the gaze behavior of participants [193], e.g., when using a visual analysis system. Here, the goal is to evaluate whether the visualizations are appropriate and effective for certain tasks, and to analyze the strategies users employ when working with the system.

CHAPTER 3

Visual Analysis of Temporal Eye Tracking Data

Temporal data, which is a specific type of sequential data, can be found almost anywhere. This makes its analysis and evaluation very important. The visual analysis of such data through interactive visualizations helps explore and understand patterns, trends, and relationships that may occur over time. Typical time series data exists, for example, in climate data, healthcare, and finance. Time series are characterized by a timestamp being associated with each data point. In the here presented analysis approaches, there lies a focus on eye tracking data. A temporal component characterizes the order of the sequence, and spatial ones indicate where a person is looking. Eye tracking data can be given as raw gaze positions or aggregated to eye movements. For more details on eye tracking and the different eye movements, see Section 2.4.

Time is continuous, nevertheless, often only discrete points or intervals are available or chosen to create visualizations for the analysis. In eye tracking, the sampling rate of the recording while acquiring raw data or the filtered eye movements (such as fixations and saccades) define such discrete data. In the analysis, it is possible to gain most insights through interactive visualization tools with multiple views that combine different visualization techniques for exploring various aspects of the data. Here, it is often possible to examine not only temporal trends but also summarized and aggregated information across time periods.

This chapter presents two visual analysis systems for eye tracking data. The first one (Section 3.1) focuses on parameter choices while filtering eye movements, and the second one (Section 3.2) on joining and comparing data from multiple sources.

3.1 Visual Exploration of Microsaccades

The first work presented in this thesis, a visual analytics system for microsaccades (Visual Microsaccades Explorer (VisME)) [10], has the goal to promote reproducibility in the data analysis of microsaccades in high-frequency eye tracking data. The visual analytics system allows users to create visualizations for microsaccade distributions and provides the possibility to interactively adjust filter settings.

This section is based on the following publication:

T. Munz, L. L. Chuang, S. Pannasch, and D. Weiskopf. VisME: Visual microsaccades explorer. *Journal of Eye Movement Research*, 12(6), 2019, doi: 10.16910/jemr.12.6.5 [10].

The images shown in this section were created with the following source code/material:

T. Munz. VisME software v1.2. Zenodo, 2019, doi: 10.5281/zenodo.3352236 [8]. (The source code is also available on GitHub: https://github.com/MunzT/VisME.)

This section presents an interactive visual analytics system (Figure 3.1) for eye tracking data, focusing on microsaccade exploration. It takes temporal raw data from eye tracking (timestamps along with gaze positions) as input. We provide different visualization and interaction techniques to visualize conventional properties of microsaccade behavior (i.e., amplitude, direction, peak velocity, duration, and temporal and spatial distribution). In an analytical process, a combination of these techniques can be used for the exploration of eye tracking datasets. To allow the detection of microsaccades, we adopted the microsaccade detection algorithm by Engbert and Kliegl [111] as a starting point and included additional features to allow for interactive parameter control. Using our system, researchers are encouraged to continuously vary these parameter values to graphically compare their impact on the dataset.

This approach deals with RQ1 and RQ2. First, it is possible to use visual analysis for sequential data. Users can use different parameter settings to process their sequential data and create visualizations to gain insights. Second, the selected parameter settings influence what is presented to the user in the visualizations and, hence, what can be discovered during the visual analysis. If the parameters

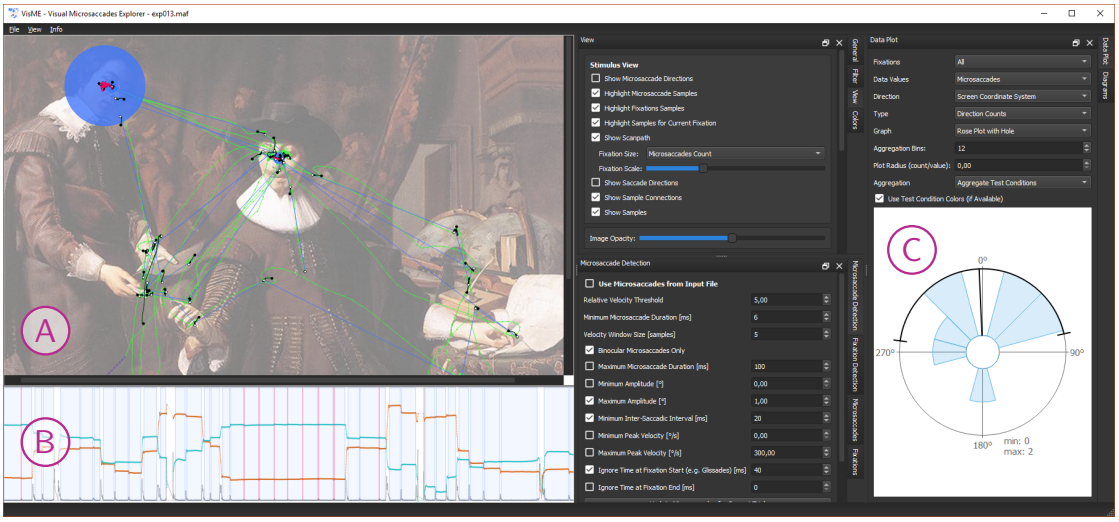


Figure 3.1 — Screenshot of the user interface showing visualizations of high-frequency eye tracking data in three linked views: (A) gaze positions with highlighted fixation and microsaccade samples on top of the stimulus, (B) temporal dependency of the eye movements and microsaccades, (C) rose plot of microsaccade directions.

are poorly selected, users may draw wrong conclusions about the microsaccades and their properties from the created visualizations.

This approach contributes by simplifying the exploration of microsaccadic datasets through interactive visual analytics. First, our system can be used to better understand how changes in the parameter values of microsaccade filters can influence the spatial and temporal distributions of microsaccades. Next, it is convenient for general visual exploration of microsaccades using interaction techniques like filtering to analyze microsaccades on different levels. In line with the increasing availability of eye movement datasets, our visual analytics system will help researchers and their reviewers critically discuss and (re-)analyze data. The source code of our implementation, a detailed description of the input and output formats, and a Python script for data preparation are publicly available [8]. Additionally, there is a more detailed manual for the system available so that the system can be readily used and the data file converter for input data can be easily adapted for other raw data.

3.1.1 Related Work

Recent years have witnessed an increase in the adoption of advanced visualization techniques by eye tracking researchers (see Blascheck et al. [60], for a

survey). However, no visualization techniques have been designed specifically for the analysis of microsaccades. Conventional visualization techniques tend to focus on low-frequency data (i.e., dwells, fixations, and saccades); for instance, scanpaths that visualize sequences of fixations and saccades or attention maps. More details about these visualization techniques can be found in Section 2.4.4. Visualizations can communicate derived statistics, which are often more informative than simply plotting the raw data, such as summary statistics of microsaccade direction and amplitude. Polar plots and rose plots are common methods to indicate the distribution of eye movement directions (e.g., Engbert and Kliegl [111], Gao and Sabel [128], Laubrock et al. [200], Lowet et al. [211], Piras et al. [253], Sinn and Engbert [291]). Scatterplots are useful in depicting the main sequence relationship between peak velocities and microsaccade amplitudes (e.g., Engbert and Kliegl [111]), and histograms illustrate the dataset's distribution of microsaccade peak velocities, magnitudes, or durations (e.g., Otero-Millan et al. [242]). Otero-Millan et al. [242] and McCamy et al. [223], for example, use figures where raw data samples are plotted on top of the stimulus, highlighting microsaccades. In order to show the temporal positions of microsaccades in relation to the eye movement, timelines are employed by McCamy et al. [224] and Otero-Millan et al. [244].

Eye tracking data are oftentimes large datasets of time series prior to feature extraction. With visualization alone, it might be difficult to handle all of the data and to fully understand it. Visual analytics [182, 308] can be an invaluable tool in allowing researchers to understand and compare complex datasets [74]; it can be a useful choice in the development of an eye tracking exploration system since different techniques can be used by people to gain insights. See Section 2.1 for some examples. For eye tracking data in general, different visual analytics methods have been investigated by Andrienko et al. [40]. Kurzhals and Weiskopf [192] introduce a visual analytics method for dynamic stimuli. To the best of our knowledge, no visual analytics system has been developed for the visual exploration of microsaccade behavior that takes eye tracking data as input and allows for the real time adaptation of filter algorithms.

Studies on microsaccadic behavior are often highly controlled psychophysical experiments (e.g., Engbert and Kliegl [111], Hafed and Clark [141], Laubrock et al. [200]), as opposed to studies that involve natural viewing tasks (c.f., Greene et al. [138], Yarbus [347]). As far as we know, no specific visualization software system exists to support the exploration of microsaccade distributions of eye movement datasets for the natural viewing of complex scenes. Our approach allows the analysis of both types of experiments. Generally, VisME is agnostic to experiment design and stimulus. This means that it can also support the comparison of datasets across different studies.

3.1.2 Requirements and Design

In this section, we provide details on the definition of microsaccades in the eye tracking domain and the requirements for the system we implemented.

We used a formative process for the development of our design, following the nested model by Munzner [231]. We focused on the outer parts of the model: domain problem characterization, data/operation abstraction design, and especially the encoding/interaction design. The process was performed in close cooperation with an eye tracking expert. In multiple sessions over a period of 16 months, our system was repeatedly refined. More details on specifying our requirements can be found in Section 2.6.

Background to Microsaccade Detection

For the automatic detection and labeling of microsaccades, different algorithms have been proposed (e.g., Bellet et al. [54], Mihali et al. [227], Otero-Millan et al. [242]), most notably by Engbert and Kliegl [111]. Nonetheless, the parameters for detecting and identifying microsaccades can vary immensely across different research studies. Furthermore, some studies (see Table 3.1 for examples) might modify basic algorithms with the introduction of additional conditional parameters (e.g., min/max amplitude, inter-saccadic interval). This lack of consistency is often tolerated in order to accommodate unavoidable variances in eye tracking data, individual behavior, and experiment designs. However, this also poses a barrier for researchers in determining whether reports of microsaccadic behavior are consistent from one study to another and the extent to which they are shaped by the chosen parameter values themselves. To facilitate the reproducibility of research results, we sought to provide a system that would support eye movement researchers in exploring and reviewing the properties of microsaccades in a given dataset as well as comparing it with another. Such a system would also serve to instruct inexperienced researchers in understanding the consequences of microsaccadic filtering.

As already described in Section 1.1, the choice of parameter values are crucial for the analysis. This also applies for the parameter settings of filter algorithms to detect eye movements [39, 61, 179]. Poletti and Rucci [254] addressed the challenges of defining microsaccades. Table 3.1 shows how the chosen parameter values and algorithmic features have differed across recent studies (i.e., 2009–2018) on microsaccades (a similar table for the period of 2004–2009 is provided by Martinez-Conde et al. [218]). In particular, Table 3.1 summarizes how the critical parameter values of the algorithm by Engbert and Kliegl [111] (e.g., λ, which is used to calculate velocity thresholds) vary across published studies. It is noteworthy that even the original values chosen by Engbert and Kliegl [111] and

Table 3.1 — Comparison of microsaccade filters used in different research projects in 2009–2018. Engbert and Kliegl [111] and Engbert and Mergenthaler [113] are added for reference, and the last row shows a summary of parameter choices.

Paper	Method	Inter-saccadic Interval	Ampli-tude	Duration	λ	Bin-ocular	Data	Other Features
Engbert and Kliegl [111]	New method	–	–	≥ 12 ms	6	yes	Eyelink System (SMI), 250 Hz	–
Engbert and Mergen-thaler [113]	Engbert and Kliegl [111]	–	$\leq 1^\circ$	≥ 6 ms	5	yes	Eyelink II, SR Research, 500 Hz	–
Dimigen et al. [106]	Engbert and Mergenthaler [113]	50 ms	$< 1^\circ$	≥ 6 ms / 3 samples	5	no	IView-X Hi-Speed 1250, SMI GmbH, 500 Hz	–
Hsieh and Tse [161]	Engbert and Kliegl [111]	80 ms	$0.15^\circ - 2^\circ$	≥ 4 samples	10	–	Eyelink2, 250 Hz	(Semi-)blinks and 400 ms before/600 ms after them removed
Pastukhov and Braun [247]	Engbert and Kliegl [111], Engbert and Kliegl [112]	–	–	–	–	yes	Eyelink 2000, SR Research, 1000 Hz	Modified algorithm to accommodate for a higher sampling rate
Mergenthaler and Engbert [226]	Engbert and Kliegl [111], Engbert and Mergenthaler [113]	30 ms	–	≥ 6 ms	3 (free viewing); 4 (fixation task)	yes	Eyelink II, SR Research, 500 Hz	–
Bonneh et al. [63]	Engbert and Kliegl [111]	–	$0.08^\circ - 2^\circ$	≥ 9 ms	–	no	iViewX Hi-Speed, SMI, 240 Hz and Eyelink II, SR Research, 1000 Hz	Raw data smoothed with a window of 15 ms; velocity range: $8^\circ - 150^\circ$/s
Benedetto et al. [55]	Salvucci and Goldberg [276]	–	$< 1^\circ$	–	–	no	SMI X-HEAD, 200 Hz	–
Otero-Millan et al. [243]	Engbert and Mergenthaler [113]	20 ms	$< 2^\circ$	–	6	yes	Eyelink 1000, SR Research, 500 Hz	–
Yokoyama et al. [349]	Engbert and Kliegl [111]	–	–	≥ 3 samples	6	yes	Eyelink CL 1000, SR Research, 500 Hz	No trials with eye blinks or eye position more than 2° away from center
Hicheur et al. [154]	Engbert and Mergenthaler [113]	25 ms	$< 1^\circ$	≥ 10 ms	4	yes	Eyelink 1000, SR Research, 1000 Hz	Microsaccades within 50 ms after a saccade were not considered as microsaccades
Pastukhov et al. [248]	Engbert and Kliegl [111]	–	$< 60'$	–	–	yes	Eyelink 2000, SR Research, 1000 Hz	Square-wave jerks
Di Stasi et al. [105]	Engbert and Kliegl [111]	20 ms	$< 1^\circ$	≥ 6 ms	6	yes	Eyelink 1000, SR Research, 500 Hz	(Semi-)blinks and 200 ms before/after them removed
McCamy et al. [222]	Engbert and Kliegl [111]	20 ms	$< 2^\circ$	≥ 6 ms	4	yes	Eyelink 1000, SR Research, 500 Hz	(Semi-)blinks and 200 ms before/after them removed
Costela et al. [94]	Engbert and Kliegl [111]	20 ms	$< 1^\circ$	≥ 6 ms	4	yes	Eyelink 1000, SR Research	(Semi-)blinks and 200 ms before/after them removed
McCamy et al. [221]	Engbert and Kliegl [111]	20 ms	$\leq 2^\circ$	≥ 6 ms	6	yes	Eyelink II, SR Research, 500 Hz	(Semi-)blinks and 200 ms before/after them removed
McCamy et al. [223]	Engbert and Kliegl [111], Engbert and Mergenthaler [113]	20 ms	$< 1^\circ$	≥ 6 ms	6	yes	Eyelink II, SR Research, 500 Hz	(Semi-)blinks and 200 ms before/after them removed
Privitera et al. [258]	Engbert and Kliegl [111]	–	$< 1.2^\circ$	–	6 and 3	no	Eyelink 1000, SR Research, 1000 Hz	First peak velocities were determined then their extend
Yuval-Greenberg et al. [355]	Engbert and Mergenthaler [113]	–	$< 1^\circ$	–	6	–	Eyelink 1000, SR Research, 1000/500 Hz	–
Fried et al. [126]	Bonneh et al. [63]	yes	$> 0.1^\circ \wedge \leq 2^\circ$	≥ 6 ms	–	–	Eyelink 1000, SR Research, 500 Hz	(Semi-)blinks and 20 ms before/after them removed; min velocity: 10°/s; peak velocity: $> 18^\circ$/s
Poletti and Rucci [254]	–	15 ms	$3' - 30'$	–	–	yes	–	–
Krejtz et al. [185]	Engbert and Kliegl [111]	–	–	≥ 6 ms	6	no	EyeLink 1000, SR Research, 500 Hz	Microsaccades detected within fixations; average position of right and left gaze positions are used
Summary	–	$0 - 80$ ms	$> 0^\circ - 0.15^\circ$; $< 1^\circ - 2^\circ$	$6 - 16$ ms	$3 - 10$	yes/no	$200 - 1000$ Hz	min/max velocity: 8°/s $- 18^\circ$/s, 150°/s; vel. window: $5 - 15$ ms; ignore time after saccade: $0 - 50$ ms; (semi-)blinks and 200 ms before/after them removed; …

Engbert and Mergenthaler [113] differed in order to accommodate for irrelevant variances across their different datasets. For example, Malinov et al. [215] detected only two eye movements as microsaccades out of 3375 saccades using a value of 0.2° as maximum amplitude. In Table 3.1, it is visible that for all experiments, larger amplitudes were used.

Eye movements that are similar to microsaccades (i.e., glissades and square-wave jerks) can be detected with the same algorithms [158] and are treated according to the researchers' discretion; also see Section 2.4.2 for further details. Given that the extraction of microsaccades can differ across different studies, it is unclear if findings can be expected to reasonably generalize from one study to another. It is often unclear why specific parameter values have been chosen or if other values might have produced different results. Thus, VisME was developed to support researchers in inspecting the influence of parameter variations.

System Requirements

We identified the following requirements for our application to explore microsaccades, serving as a basis for the visual encoding and interaction design.

Ability to:

- explore microsaccades in the context of the entire eye tracking data in space and time.
- explore the relationship between space and time for microsaccades.
- explore microsaccadic properties.
- explore individuals and groups of participants.
- explore the location of microsaccades within fixations.
- change parameters for microsaccade detection.
- study statistical values when changing parameters.
- differentiate between microsaccades and similar eye movements.
- explore the influence of fixation filters on microsaccades.
- explore the relationship between microsaccades and test conditions.
- integrate VisME into existing analysis pipelines.

We believe that interactive visualization, along with data filtering, is the most appropriate approach to support scientists exploring these aspects in a visual analytics system.

3.1.3 Visual Analytics Approach

In this section, we detail the implemented filters for microsaccades and fixations, and the different visualization and interaction techniques employed to explore high-frequency eye tracking data. For the analysis, fixations have to be determined first. Afterward, microsaccades can be detected within their time ranges, and different views can be used for visualization and interaction. The general user interface design can be seen in Figure 3.1.

Eye Movement Filters

Eye tracking data can be grouped into different eye movement classes. Often, these movements can be precalculated by the eye tracking system itself. In VisME, it is possible to explore eye movements (fixations and microsaccades) that were determined in preprocessing or with the system itself. In the following paragraphs, we describe the filters we use in our application to interactively label microsaccades and fixations. A more detailed description about eye movement filters in general can be found in Section 2.4.3.

Microsaccade Filter – In contrast to detecting saccades and fixations, the software of eye trackers do usually not provide filters for microsaccades. For interactive microsaccade detection, we chose the velocity-threshold algorithm by Engbert and Kliegl [111] as an example; other algorithms could have been used as well. A free parameter λ is used for a velocity threshold in 2D space, and a minimum microsaccade duration is also fixed. We decided upon these parameters as they were prominently mentioned in previous work (see Table 3.1).

Available parameters for microsaccade detection in our system are:

- λ for the velocity threshold
- minimum and maximum duration
- minimum and maximum amplitude
- minimum and maximum peak velocity
- velocity window size
- time being ignored at beginning/end of fixations (e.g., to ignore glissades)
- time being ignored after a microsaccade, i.e., minimum inter-saccadic interval (to ignore over-shoots)
- time being ignored before/after missing data
- monocular or binocular microsaccades

Initially, we set the parameters in our system according to the values mentioned by Engbert and Mergenthaler [113] ($\lambda = 5$, minimum duration: 6 ms, velocity

window size: 5 samples, binocular microsaccade detection, maximum amplitude: 1°) and set minimum inter-saccadic interval to 20 ms and ignored the first 20 ms of fixations.

Fixation/Saccade Filter – Our method uses the same algorithm that we implemented for detecting microsaccades but with different default parameter values ($\lambda = 8$, minimum saccade duration: 3 ms, velocity window size: 9 samples, minimum saccade amplitude: 1°, minimum inter-saccadic interval: 50 ms). This method is a saccade filter, and fixations are defined as the intervals between two saccades. Mergenthaler and Engbert [226], Otero-Millan et al. [242], Sinn and Engbert [290], and Laubrock et al. [201] also used this algorithm to detect saccades. When applying different fixation filters or parameter values, it is possible that some fixations are not detected at all or multiple fixations are detected as only one fixation. While one algorithm detects two fixations connected by a saccade, another might detect just one fixation that contains a microsaccade. This, of course, influences the relationship between microsaccades and fixations. Also, if more or fewer data samples are included within a fixation, the center of fixation shifts.

Visualizations

To explore the distribution of microsaccades, our application provides multiple linked views [273] that can be explored interactively: A stimulus view, which shows the stimulus, the raw data, and a scanpath visualization (Figure 3.1 A); a timeline view for visualizing the temporal distribution of microsaccades in relation to the eye movement (Figure 3.1 B); data plots used for visualizing descriptive statistics and details on microsaccade distribution (Figure 3.1 C); and histograms and scatterplots for further microsaccadic properties.

Stimulus View – The stimulus view provides an overview of the eye tracking data. In this view, all raw eye tracking samples, fixation samples, microsaccade samples, and missing data ranges can be shown or highlighted (Figure 3.2 (a)). A plot with directional microsaccade distributions can be displayed with lines that connect the start and end samples of microsaccades colored with a gradient to encode the directions and locations of the microsaccades on the stimulus (Figure 3.2 (b)). A common scanpath visualization for fixations and saccades is also available (Figure 3.2 (c)). The size of the fixations can be either in relation to the duration of fixations, the number of microsaccades within fixations, or in such a way that each circle has the same size. Using filters, it can be decided which information shall be visible in this view.

Timeline – The timeline (Figure 3.1 B and Figure 3.3) gives a better understanding of the relationship between time, eye positions, and eye events. It is visible

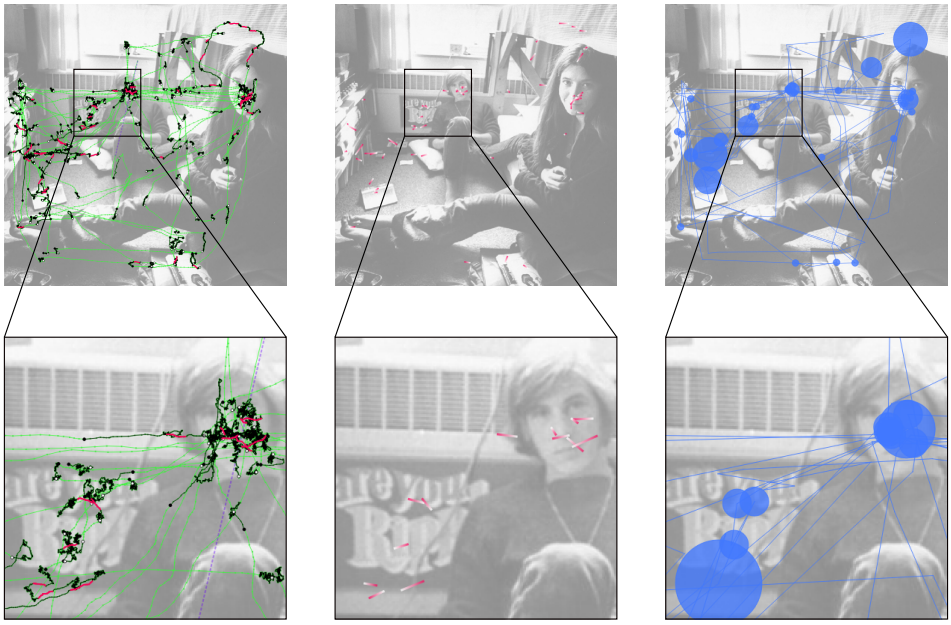


Figure 3.2 — Different aspects of the eye tracking data can be visualized in the stimulus view. (a) Connected raw samples (green), connected fixation samples (black), connected microsaccade samples (pink), and missing data ranges (purple). (b) Directional microsaccade lines: pink for the start sample and white for the end sample. (c) Scanpath (blue) with fixations represented by circles (their size is in relation to the number of microsaccades within them) and saccades by lines.

where fixations are located, and microsaccades appear. As some experiments depend on specific temporal events, event positions can be highlighted as well. This view is zoomable to see more details of the surroundings of microsaccades and fixations.

Data Plots – In a separate section of our system, descriptive statistical graphics (e.g., rose plots) and further fixation-related visualizations are used to show more details on the dataset in relation to microsaccades and the chosen parameters. Rose plots (see Figure 3.1 C) or polar plots are used to show the direction of microsaccades; the data is aggregated to a specified number of bins (if not mentioned otherwise, 12 bins are used in all graphics). If multiple test conditions are specified, data of each test condition type can be visualized with another color value. Additionally, the mean direction and standard deviation are shown in black. To see the microsaccade distributions, especially the number, and locations within fixations, the plot type can be changed; see Figure 3.4 (a) for an example. In this image, all fixations of a trial are plotted on

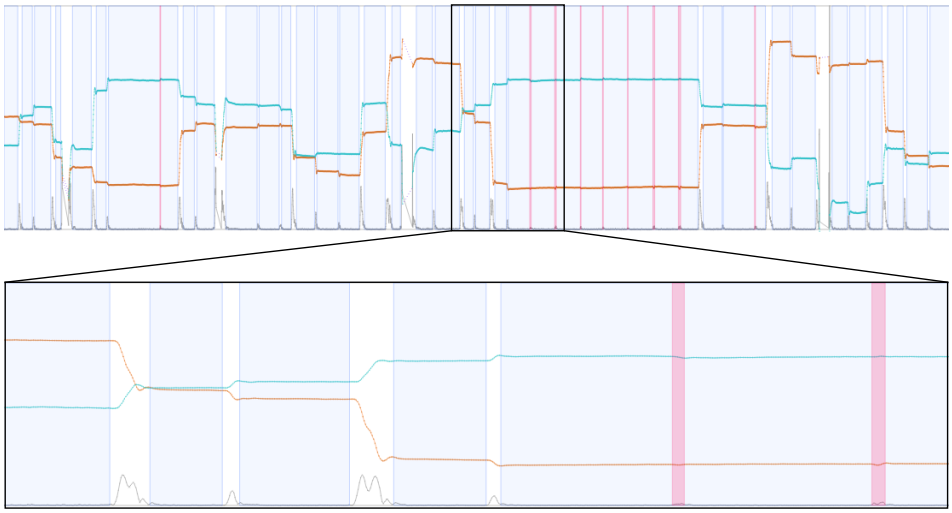
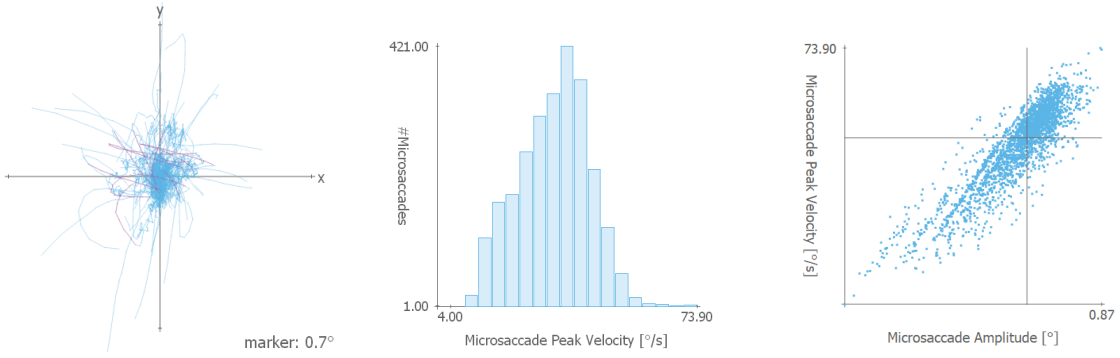


Figure 3.3 — Timeline view: timeline (top) and zoomed timeline (bottom) to explore more details. Timelines show the eye positions in the x (orange) and y (turquoise) direction, the velocity (gray) determined by these positions, missing data ranges (purple) in the x and y direction, fixation areas (blue), and microsaccade locations (pink).



(a) (b) (c)

Figure 3.4 — (a) Position of microsaccades relative to fixations of a trial in the context of the fixations that are plotted on top of each other (microsaccades are highlighted in pink). The marker value indicates the extent of the marked position on the x and y axes; values are measured in visual degree. (b) Histogram for microsaccade peak velocity and (c) the relationship between amplitude and peak velocity as scatterplot using a logarithmic scale for the *painting* dataset (see Section 3.1.4).

top of each other; microsaccades are highlighted in pink. A similar plot that can be created with VisME shows only microsaccade directions in relation to the fixation center. The same start and end points of microsaccades are used as

shown in the image, and a color gradient is applied to indicate their direction, similar to the direction plots in the stimulus view (Figure 3.2 (b)).

While all the data visualized in this area can be shown with microsaccade directions in screen coordinates, it is also possible to transform the data in such a way that the directions of microsaccades are rotated toward the next fixation. For these plots, the direction to the next fixation points to the top (e.g., 0° in rose plots). This transformation is useful for a better understanding of microsaccades and the whole eye movement. To determine the relationship toward the next fixation, a direction vector between two fixation centers is used. For the rose plots, a microsaccade within the first fixation is translated in such a way that its start point is located at the center of the corresponding fixation. The angle between the vector of a microsaccade (from start to end sample) and the direction vector determines the angle used for the direction. For the other data plots, the direction vector determines how all data of a fixation has to be rotated.

Histograms and Scatterplots – A further area of the system provides the possibility to explore the temporal locations, durations, amplitudes, and peak velocities of microsaccades in histograms and scatterplots (Figure 3.4 (b) and (c)). Especially scatterplots that show the relationship between amplitude and peak velocity are commonly used for both saccades and microsaccades to explore the main sequence. Additionally, histograms show the temporal distribution of microsaccades within fixations.

Interaction Techniques

Our visual analytics system provides many interaction methods to explore the raw data on different levels, i.e., for fixations, participants, trials, and test conditions. Interaction supports the analysis process for microsaccades, for which we will provide some examples in the demonstration of some use cases (see Section 3.1.4). With brushing and linking, a combined perspective of different aspects of the data shown in different views can be obtained. The parameters for detecting microsaccades can be modified, the visible data can be filtered on different levels, and standard navigation techniques such as zooming and panning are available. It is possible to select which eye data (left/right/averaged) should be used for analysis, and a rubber band is available to select a time range in the timeline. The filters allow exploration of the data on different levels: for participants, trials, and test conditions. Furthermore, the amount of data visualized in the different views can be adapted by hiding different elements. As all views are linked, it is possible to select a fixation (if a single trial is being explored) in either time, space, or a list of all fixations to highlight this fixation; it will be highlighted in the other views as well, the corresponding

data plots will be shown, and some details on the fixation will be displayed. Additionally, it is possible to walk through the scanpath for every fixation. As it is also of interest to see relationships of sequential fixations, it can be specified how many neighboring fixations shall be visualized, and the data plots will include these fixations as well. Moreover, some statistical values about the selected trials are updated when parameters change. These include information about fixations (e.g., count, duration, or percentage containing microsaccades) and microsaccades (e.g., count, duration, amplitude, peak velocity, count per second/fixation, or inter-saccadic interval).

Data Import and Export

Our application uses its own format for input data to be independent of any eye tracker. High-frequency eye tracking data is required as input, with a minimum frequency of 200 Hz in order to detect microsaccades [158]. For each participant, a separate file is required that can contain multiple trials specifying raw samples, fixations, microsaccades, and event locations; a second file type can contain information about test conditions (e.g., tasks or other trial-specific circumstances). Files with the current (possibly calculated) eye movement data (raw data, fixations, microsaccades, and events) can be exported for further analysis with other statistical software such as R, Python [321], or MATLAB and for later import into VisME. Additionally, aggregated statistics for different properties of fixations and microsaccades containing data for each participant and test condition can be exported for analysis in other applications. More details about the file formats used in VisME are available by Munz [8].

Our system can be integrated into full study and analysis pipelines. Researchers have to convert their eye tracking data into our expected input format. It is possible to import detected microsaccades determined in other steps of their analysis to verify their properties or to calculate a new set of microsaccades with VisME by visually inspecting the result. Our system can then be used to explore the data and to export the raw data along with detected microsaccades as well as aggregated data for further processing.

Implementation Details

Our application is platform-independent and was tested on Windows 10 and Linux. It is implemented in C++ with Qt 5.9 using the Graphics View Framework for interactive visualizations. We reimplemented the R code available by Engbert et al. [114] to detect microsaccades and added some additional conditions as described previously in Section 3.1.3.

In the preprocessing steps for the demonstration of use cases and the user study, movements recorded by an EyeLink eye tracker were first converted from .edf files to .asc files using the converter available by SR Research: the EyeLink EDF2ASC Converter. Then, we used a script written in Python to create an input file in the format expected by our application, containing eye tracking positions and fixations. The eye positions of our raw data were originally given in screen coordinates. Using the formula given in Section 2.4.1, we converted them into visual angles for further processing in VisME.

In our system, only the microsaccade filter, as described before, is implemented, but it is designed so that other methods can be implemented and added with little effort. In order to explore microsaccades detected with other algorithms, the precalculated data can also be imported into our system.

3.1.4 Evaluation

To demonstrate how our application can be used, we use externally collected high-frequency eye tracking data from two independent experiments to show use cases for our approach and in a usability study.

Eye Movement Dataset

In both experiments that were conducted to collect the two datasets, participants were asked to look at images with some given task for the exploration. Eye movements were recorded with remote eye trackers (Eye-link 1000, SR Research) using a chin rest to stabilize the heads of participants. More details about this type of eye tracker and its properties can be found in Section 2.4.1. Fixations were determined by the system's eye tracking software. While monocular data is available for the first dataset, the second dataset allows binocular analysis as well.

As a first dataset (*photo* dataset), we use data collected from experiment 3 of Greene et al. [138]. It contains eye tracking data on 16 participants who viewed 20 grayscale natural images for 60 seconds whilst performing one of four tasks. The tasks were: memorize the picture (*memory*), identify the decade in which the picture was taken (*decade*), assess how well the people in the picture know each other (*people*), and determine the wealth of the people in the picture (*wealth*). The data was recorded at 1000 Hz, and only the right eye was tracked. The dataset was originally created to verify Yarbus's assumption that the eye movement is highly influenced by an observer's task. We chose this dataset because it is high-frequency data that allows extraction of microsaccades, and participants performed different high-level cognitive tasks that were likely to have engaged covert attention even if the study was not designed to investigate this aspect of

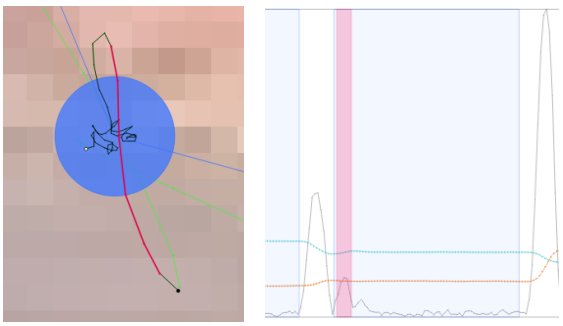
gaze behavior. Many researchers believe that covert attention can influence the frequency of microsaccades and their directions (e.g., Hafed and Clark [141], and Hermens and Walker [149]; also see Section 2.4.2).

The second dataset (*painting* dataset) was recorded to explore the occurrence of microsaccades in free-viewing conditions. It is available at 500 Hz for both eyes. Gaze samples for averaged eye movements were determined as the mean value of right and left eye positions and fixations as the maximum fixation areas of both eyes. Twenty participants looked at 60 randomized colored paintings showing multiple persons for about 15 seconds per image. The participants' task was to pay attention to the presented people, their mood, and relationship to each other (*acquaintance*), and they had to answer questions afterward.

Use Cases

We demonstrate how it is possible to visually explore the directions and distributions of microsaccades with VisME. For this exploration, we use the initial parameter values mentioned before in Section 3.1.3.

Detecting Glissades – Glissades are eye movements that immediately succeed saccades (see Section 2.4.2). In order to demonstrate the confusability of microsaccades and glissades at the start of fixation periods, we changed the potential glissade duration to 0 ms; this is an adjustable parameter in VisME. Subsequently, we inspected the stimulus view and timeline with highlighted microsaccades of one trial of the *painting* dataset. We noticed that many detected microsaccades were both spatially and temporally located at the beginning of fixations, right after saccades. Velocity peaks also indicated that they might be microsaccades. However, as they are located right after saccades, they are more likely to be glissades (see Figure 3.5). Additionally, we had a look at the histogram for the temporal location of microsaccades within fixations (Figure 3.6 (a) and (b)): it is visible that for both the current trial and the whole dataset, there is a high peak for potential microsaccades within the first 40 ms, which might be glissades instead. This suggests that this phenomenon is present in the whole dataset. Thus, we modified the parameter for the potential glissade duration from 0 ms to 40 ms for the *painting* dataset for the remaining part of the analysis. Note that this parameter would be arbitrarily determined by most researchers with no opportunity for reanalysis by others in the absence of a tool like VisME. In Figure 3.6 (c) and (d), the early fixation period was excluded from the analysis, which resulted in a more suitable distribution of detected microsaccades. There is still a peak in the second bin (Figure 3.6 (d)) but visibly diminished relative to the previous first bin. This could explain why Hicheur et al. [154] did not consider microsaccades within even 50 ms after a saccade.



(a) (b)

Figure 3.5 — Eye movement is detected as microsaccade (pink); it can be identified as glissade on the stimulus and in the timeline: (a) Connected samples that belong to the microsaccade are located at the beginning of the fixation (black). (b) Zoomed view of the timeline: the detected microsaccade is located at the beginning of a fixation (blue) right after a saccade. (x values: orange, y values: turquoise, velocity: gray). Example used from the *painting* dataset.

We noticed this behavior in both datasets; for the *photo* dataset, we chose a threshold of 20 ms to obtain a similar distribution. Depending on the chosen fixation filter and possibly other reasons we are not aware of, this value might have to be chosen differently for other datasets. Our system can be used to find a suitable minimum time range to remove glissades without removing actual microsaccades from further analysis.

Microsaccade Directions for Test Conditions – To visualize the microsaccade directions across different tasks, rose plots were employed to illustrate, for all available trials, the distribution of microsaccades in every direction (Figure 3.7). This reveals potential asymmetry in the circular distribution of microsaccadic movements. We use both datasets and differentiate trials for the *photo* dataset by different tasks, and for the *painting* dataset, we visualize all data together. In the first row, the *photo* dataset reveals a tendency for microsaccades to be oriented horizontally and vertically within the image. For the *painting* dataset, microsaccadic movements were biased toward the top, which could be indicative of a recording bias. The rose plots in the second row illustrate the directions of microsaccades relative to the next fixation (toward the next fixation means to the top of the graph). The *photo* dataset revealed a tendency of microsaccades toward the next fixation (especially for the task *people*), indicating that microsaccades predict the next fixation. Additionally, there is also a strong tendency toward the opposite direction. For the *painting* dataset, a similar vertical bias was visible that was less pronounced and favored the opposite direction to the next fixation. In these plots, no correlation between the two tasks related to people (*people* and *acquaintance*) is visible, but a similarity of data recorded in the same experiment can be seen. A possible reason why these

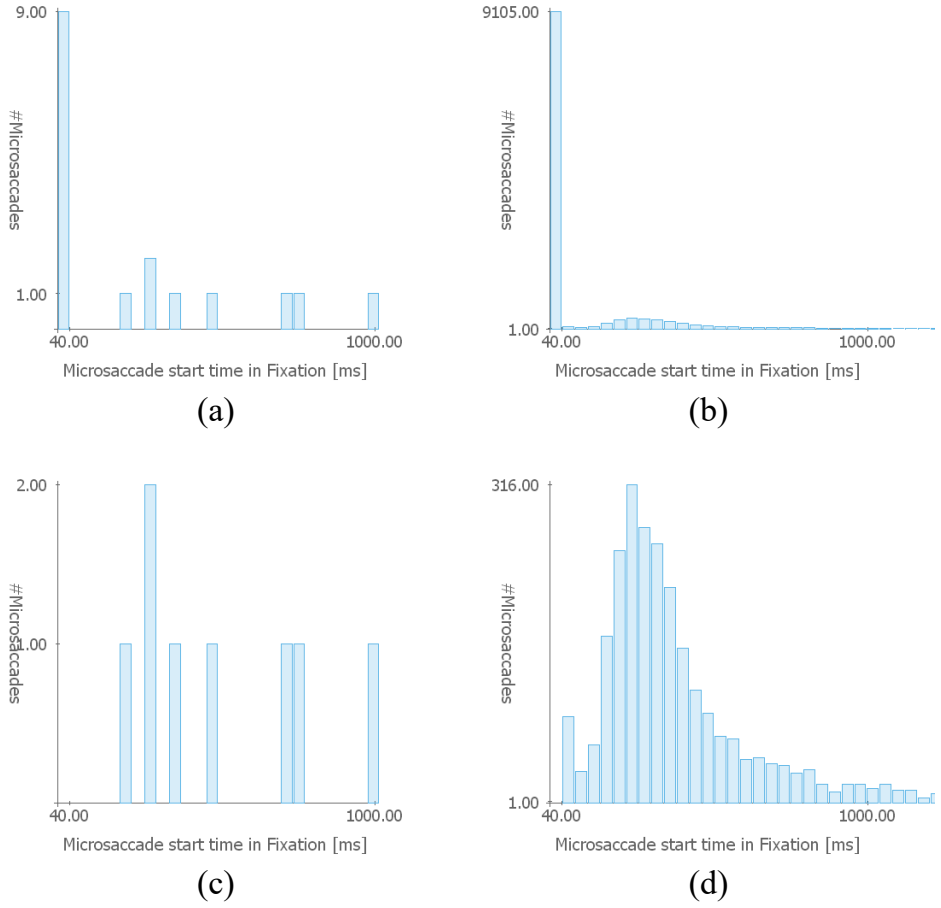


Figure 3.6 — Temporal positions of detected microsaccades as a histogram for (a) one trial of one participant and (b) all trials of all participants in the *painting* dataset (bin size: 40 ms). A high peak in the first bin indicates that many glissades (over-shoots right after saccades, here in the subsequent 40 ms) were detected as microsaccades. Figures (c) and (d) show the same data when all microsaccades within the first 40 ms of a fixation are ignored from the analysis.

different results are obtained might be due to data quality and existing noise when using monocular data.

Microsaccade Directions for Participants – Our next step was to examine the data on an individual level for different participants. For selected participants viewing the *painting* dataset, images are visible in Figure 3.8. It is possible to see how data can vary between different participants: the images show results for participants that might indicate that microsaccades move toward the next fixation, the opposite direction, or in arbitrary directions. Such visualizations could support researchers in identifying different microsaccade patterns available

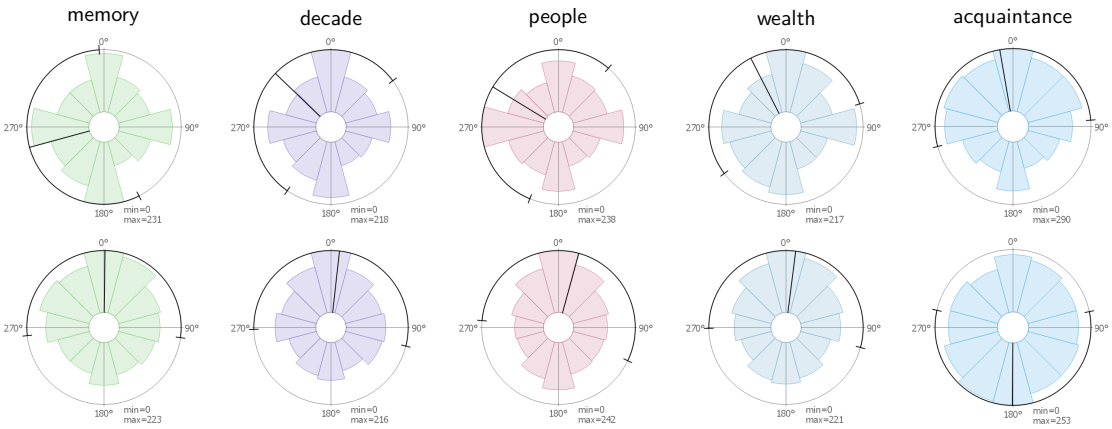


Figure 3.7 — Rose plots for microsaccade distributions for the four different tasks of the *photo* dataset (*memory*, *decade*, *people*, and *wealth*) and the *painting* dataset (*acquaintance*). First row: directions of microsaccades in screen coordinates as visible on the stimulus. Second row: microsaccade directions in local coordinates: they are rotated in such a way that 0° means that a microsaccade is in the direction to the next fixation. Min and max specify the values in the middle of the plot and on the outer circle, respectively; values are measured in microsaccade count.

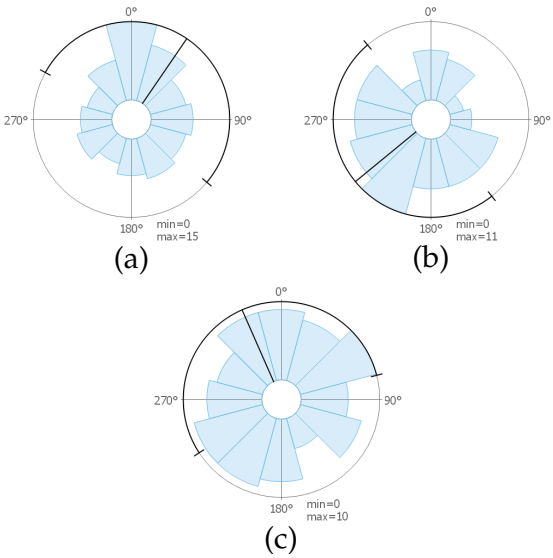
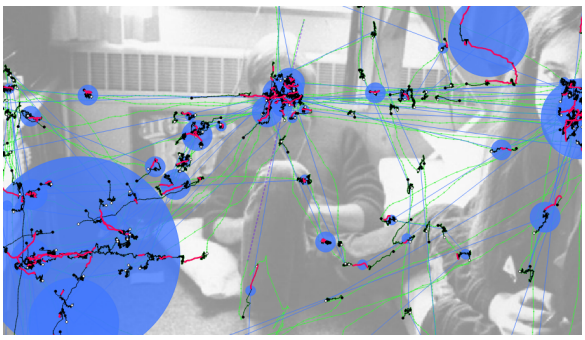


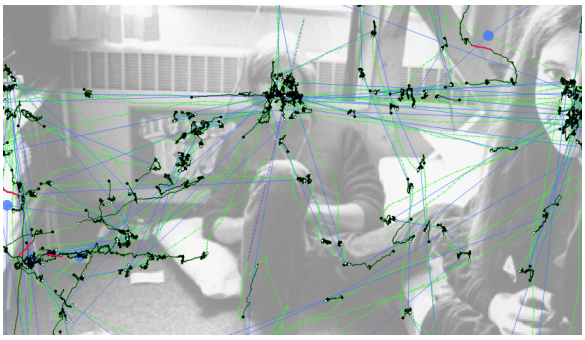
Figure 3.8 — Rose plots showing microsaccade directions rotated toward the next fixation for a selection of participants from the *painting* dataset. Participants whose microsaccades have a direction that (a) tends to be toward the next fixation, (b) toward the opposite direction, and (c) that are equally distributed toward each direction.

for certain participants and compare if patterns are similar to the aggregated directional distribution of all participants (see Figure 3.7 (*acquaintance*, bottom)) or rather outliers.

Microsaccade Detection for Changed Parameters – In order to see the strong influence of parameters on the detection of microsaccades, we used two different



(a)



(b)

Figure 3.9 — (a) Result of the microsaccade filter when changing all parameter values in such a way that more microsaccades can be detected using values in the range as given in Table 3.1 and (b) when using parameter values to limit the detection. A trial from the *photo* dataset is shown.

settings of parameter values taken from the value ranges visible in Table 3.1: the first settings contain values that result in few microsaccades (minimum intersaccadic interval: 80 ms, maximum amplitude: 1°, minimum duration: 12 ms, $\lambda = 8$, velocity window: 5 samples, ignore time at beginning of fixations: 50 ms; parameters not mentioned were deactivated) and the other ones in many (maximum amplitude: 2°, minimum duration: 6 ms, $\lambda = 3$, velocity window: 5 samples). The first parameter settings missed many actual microsaccades, and the second one detected many false microsaccades. The difference in the number of detected microsaccades varied for the trial visible in Figure 3.9 from only 4 to 184 (if glissades were excluded (here, the first 20 ms of a fixation), still 138). It is apparent that there is a large range of possible sets of microsaccades that can be detected with values chosen between these two different settings. It clearly shows that, as introduced in Section 1.1 in the context of RQ2, it is important to select appropriate parameter values for the analysis. The visual analytics system can help in exploring the data to choose appropriate values for a given dataset. A visual inspection of the stimulus view is necessary to confirm if the parameter changes improve the detection of microsaccades or result in false positives or missed microsaccades.

To further explore the influence of parameter choices, we explored changes of individual parameter values to the maximum or minimum values given in

Table 3.1. We created aggregated statistical distributions for all participants and trials to show how they change when adapting parameter values; visualizations of the results can be found in the supplemental material of the corresponding publication [10]. In the following, we compare the results of our default settings to some adaptions. We can see a strong influence for different velocity threshold (λ) values. Mergenthaler and Engbert [226] use $\lambda = 3$ for freeviewing experiments. Using the same value for our datasets shows an increase in the number of detected microsaccades by a factor of three. The main direction changes toward the bottom and more microsaccades are detected with smaller velocities, amplitude, and duration. Some of them are also located at a later position during a long fixation. When using $\lambda = 10$ [161] instead, fewer microsaccades are detected, which are mostly moving to the left and right side. The number of microsaccades is reduced for small peak velocities, high duration, and small amplitudes. When the threshold for the minimum duration is increased, fewer microsaccades with smaller duration and smaller amplitude are detected. Changing the size of the velocity window [63] shows only for the *painting* dataset a noticeable different pattern. In the main sequence, we see microsaccades with very large velocity values; this might be an indication of noise. The other plots also show slightly different patterns (e.g., the duration histogram seems to be shifted). As the *painting* dataset was recorded for both eyes, we can compare the influence of using monocular or binocular detection. Using monocular detection, more microsaccades are detected. There is an increase in microsaccades with higher peak velocity, smaller duration, and smaller amplitude. We can also see that many microsaccades have a direction toward the bottom of the screen.

Influence of Fixation Filters – As the fixations themselves also influence microsaccades, we used the fixation filter of our system (see Section 3.1.3 for details) to detect different fixations than the ones already determined by the software of the eye tracker. In Figure 3.10, the raw data with highlighted fixation and microsaccade samples is visible for an example scene. The left images were created for the fixations provided by the system's eye tracking software, and the right ones were created for the fixation filter of VisME. It is visible that the fixations change most notably in size, and hence, the microsaccade amount and directional patterns change as well.

In our exploration process, we observed that microsaccade distributions depended on different factors. Fixation labeling defines the data that is used for the extraction of microsaccades and defines the spatial relationship between microsaccades and the next fixation. Microsaccade parameters are important for the calculation of microsaccades within these fixations. They influence, for example, if confusable eye movements (e.g., glissades) will be regarded as

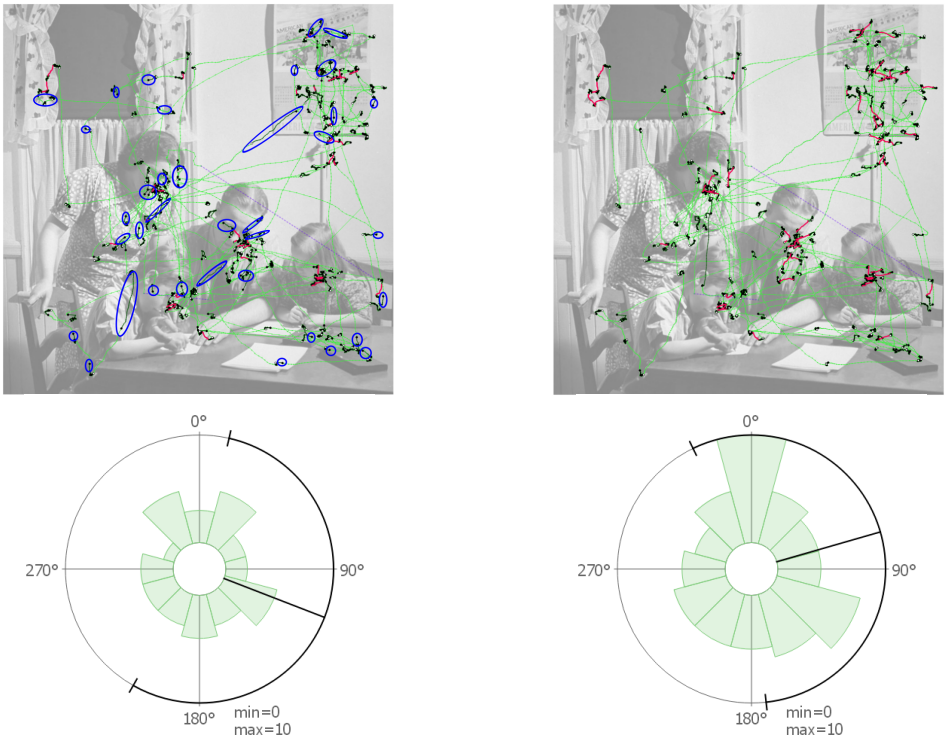


Figure 3.10 — The usage of another fixation filter influences the sample ranges used for microsaccade detection and, thus, the rose plot visualizations. The left images use the fixations from the eye tracking software, and the right ones use the fixations calculated by VisME. Microsaccade detection was performed in both cases using the microsaccades filter with default parameter values. In the images at the top, some differences in the detected fixations are highlighted in blue. In the images at the bottom, it is visible that the microsaccade distribution for the trial changes as well.

microsaccades, which has a strong influence on the overall statistics. By looking at individual trials, it is possible to determine if small microsaccades were missed or if fixation samples might have been mislabeled as microsaccades. Overall, it is possible to explore interactively the effects of parameter value changes using all previously mentioned visualizations and further statistical values provided by the system. For instance, for unknown datasets, it is possible to verify whether the parameter choices for microsaccades exploration were appropriate. Additionally, we showed that our system can handle both monocular and binocular eye movement analysis, and it is not limited to one type only. It is up to the researcher to handle noise in preprocessing and to verify if the detected microsaccades are correct. Aligned with our idea to support reproducibility, critical readers of research reports who want to verify the plausibility

of suggested research results benefit from such a system. While there is an increase in sharing raw data when publishing research, it additionally requires a software system to reanalyze the data or reproduce the results. With VisME, researchers will be able to check for themselves if applied parameter settings or filters are appropriate or if there might be other settings that would better fit the data.

Usability Study

We performed a usability study with a think-aloud protocol analysis [117] to collect qualitative feedback (see Section 2.7) on the usefulness and usability of VisME for eye movement researchers. First, participants completed a questionnaire on their research background. Next, we introduced them to our system by explaining VisME's main features. Subsequently, participants used the system to explore the *painting* dataset of our use cases. We provided a list of tasks to guide the participants' use of the system and to support their awareness of available features. Nonetheless, participants were also encouraged to freely explore the use of the system for whichever aspects interested them. The participants were able to explore how adjusting different parameter values influenced derived microsaccadic statistics by observing how doing so impacted the different visualizations. Some participants asked for specific features and if they were available in the system; we demonstrated to them that such features were already implemented. Finally, the participants were asked to complete a standardized questionnaire on usability [71] that was extended to include some eye tracking specific questions. Each question was rated on a Likert scale from 1 (strongly disagree) to 5 (strongly agree). Participants could provide additional text feedback and suggestions for future improvements.

We conducted the user study with 12 voluntary and independent eye tracking experts (three women and nine men; 11 participants were between 20 and 39 years old, and one was at least 50 years old) who are not authors of the corresponding paper [10]. Our participants' backgrounds include computer science, physics, and neuroscience. We wanted to have eye tracking experts with different research goals, especially those who work with microsaccades. The eye tracking researchers belong to four different research groups with specializations in different eye tracking and eye movement related areas. These areas include the analysis of microsaccades, the relationship between eye movements and neuroscience, the development of medical devices to enhance vision, and the development of new eye movement-related algorithms. Three of our participants had more than five years of experience with eye tracking, four between three and five years, and five less than three years. Eight of the researchers rated their eye tracking experience from 4 (high) to 5 (very high) on a scale from 1 to

5 (mean: 3.9). Proficiency with microsaccades varied among participants: Four stated to have high and very high experience (4 and 5), whereas six stated to have little and very little experience (1 and 2); the mean value was 3. More than half of all participants claimed to use visualizations often in their analysis.

Before starting the actual experiment, we asked the participants about their understanding of microsaccades and how they would filter for them. Depending on their proficiency with microsaccades, they could provide detailed information. Most researchers mentioned that microsaccades are located within fixations. The maximum amplitude of microsaccades was stated as ranging from 0.5° up to 2° by different participants, but most of them employ a threshold of 1°. Furthermore, some mentioned that a manual inspection of each detected microsaccade is very important. For the detection of microsaccades, the algorithm by Engbert and Kliegl [111] was named by a few researchers, which confirms that our initial algorithm choice was appropriate. For some researchers, it is very important that the detection is done on binocular data, while others use monocular data. This also shows that it is important to support both types of data analysis in our system. Additionally, we asked participants how they would process eye tracking data to explore microsaccade distributions. Most of them described similar approaches to the one we realized in VisME. Additionally, they would verify the data quality at the beginning. Most participants would use MATLAB as analysis software.

All participants liked our system, were able to use it without any problems, and gave a lot of positive feedback. They agreed that our system could be beneficial for teaching purposes (mean: 4.9) and that the eye tracking community would benefit from such a tool (mean: 4.3). The question of whether participants would prefer to use this software over the steps they described initially for exploring microsaccades had a mean value of 3 and the highest standard deviation. This roughly correlates with the proficiency with microsaccades analysis. Researchers who have less experience were more likely to choose this software. All researchers agreed that VisME served its intended purpose of rendering the analysis of microsaccades more reliable and transparent.

In particular, the interactive and visual features of our system were received positively. The possibility to click on fixations that are then highlighted in both space and time and the possibility to scroll through trials were highlighted as a preferred feature. Many researchers stated that VisME is especially useful to get a quick overview of the data.

Recommendations for additional features varied widely depending on the participants' research background and interests. For a system supporting the whole analysis process, participants suggested that our system should be able

to deal with preprocessing steps like data smoothing or processing of eye blinks as well; currently, preprocessing of raw data has to be performed externally. Furthermore, manual inspection of microsaccades is very important. Therefore, manual correction of both fixation and microsaccade areas would be required to adapt detected eye movements that were not marked correctly. Additionally, analysis related to specific events is very important, especially in controlled experiments. In our system, it is possible to visualize the temporal positions of events, but it is not possible to consider temporal aspects related to the events in the analysis. Currently, it is possible to analyze microsaccade directions in relation to neighboring fixations; two participants asked for an extension to allow this analysis to be performed in relation to arbitrary target positions on the stimulus (e.g., the center of an image). As there are many different approaches available to detect microsaccades, some participants wished for further algorithms to be supported.

3.1.5 Conclusion

We present a visual analytics system for exploring eye tracking data, with a focus on microsaccade analysis. With this system, eye movement researchers are able to explore and understand microsaccade distributions in space and time. In particular, the interactive nature of the visualizations, namely the ability to vary multiple parameter values, allows researchers to determine how sensitive their findings are to parameter variations. This system is tailored for research purposes in that it allows researchers to analyze microsaccadic patterns on the level of participants, trials, and test conditions. It also allows for flexible adjustments of parametric values across these levels in order to account for huge individual differences, if necessary.

Our system allows for more transparent discourse between researchers and increase the value of public datasets; it supports reproducibility and promotes open research. Eye movement researchers are able to decide for themselves if appropriate parameter values were used, as well as to discover unexpected eye movement behavior or verify novel hypotheses on old data. Our system is especially helpful in getting an overview of available datasets and provides a simple approach to exploring microsaccades for researchers with little experience. In addition, it can serve as an instruction system to help researchers better understand microsaccade movements and issues in their detection. While there remain many possible additions to the application (see user feedback in the previous section), we believe that eye movement researchers profit from its visual analytics features to explore microsaccades.

3.2 Comparative Gaze Analysis

In this section, a visual analysis approach (Eye Tracking Fusion System (ETFuse)) [12, 13] is presented to explore eye movement data from two people playing competitive and collaborative virtual board games. While the previously presented system focuses on analyzing eye tracking data from one participant at a time and an aggregated analysis for a larger group, this approach allows a detailed comparison of the eye movement of two people. The previous approach mainly focused on detecting and analyzing eye movements, and this approach focuses on joining data streams from different sources for their analysis.

This section is based on the following publications:

- T. Munz, N. Schäfer, T. Blascheck, K. Kurzhals, E. Zhang, and D. Weiskopf. Comparative visual gaze analysis for virtual board games. In *Proceedings of the 13th International Symposium on Visual Information Communication and Interaction (VINCI '20)*, article 11, pages 1–8. Association for Computing Machinery, 2020, doi: 10.1145/3430036.3430038 [12].
- T. Munz, N. Schäfer, T. Blascheck, K. Kurzhals, E. Zhang, and D. Weiskopf. Demo of a visual gaze analysis system for virtual board games. In *ACM Symposium on Eye Tracking Research and Applications (ETRA '20 Adjunct)*, article 2, pages 1–3. Association for Computing Machinery, 2020, doi: 10.1145/3379157.3391985 [13].

The images shown in this section were created with the following source code/material:

T. Munz, N. Schäfer, T. Blascheck, K. Kurzhals, E. Zhang, and D. Weiskopf. Supplemental material for comparative visual gaze analysis for virtual board games. DaRUS, V1, 2020, doi: 10.18419/darus-1130 [11]. (The source code is also available on GitHub: https://github.com/MunzT/ETFuse.)

This section introduces a visual analysis approach (Figure 3.11) for the combined gaze analysis of two players competing against each other in a web-based game. The fact that eye movement data of players involved in the same online game is usually recorded on separate computers, perhaps using different eye tracking devices with different sampling frequencies, is challenging for comparative analysis. Synchronizing and comparing such eye movement data can be difficult and time-consuming. In our approach, the eye movement data of two players recorded on different devices can be synchronized and visually represented for analysis purposes. As input, it uses the eye movements (fixations and saccades) collected with two different eye trackers and detected with their corresponding software, video streams, mouse click events, and possibly additional temporal events. We demonstrate the effectiveness of our approach by applying it to an online version of the board game Go [125].

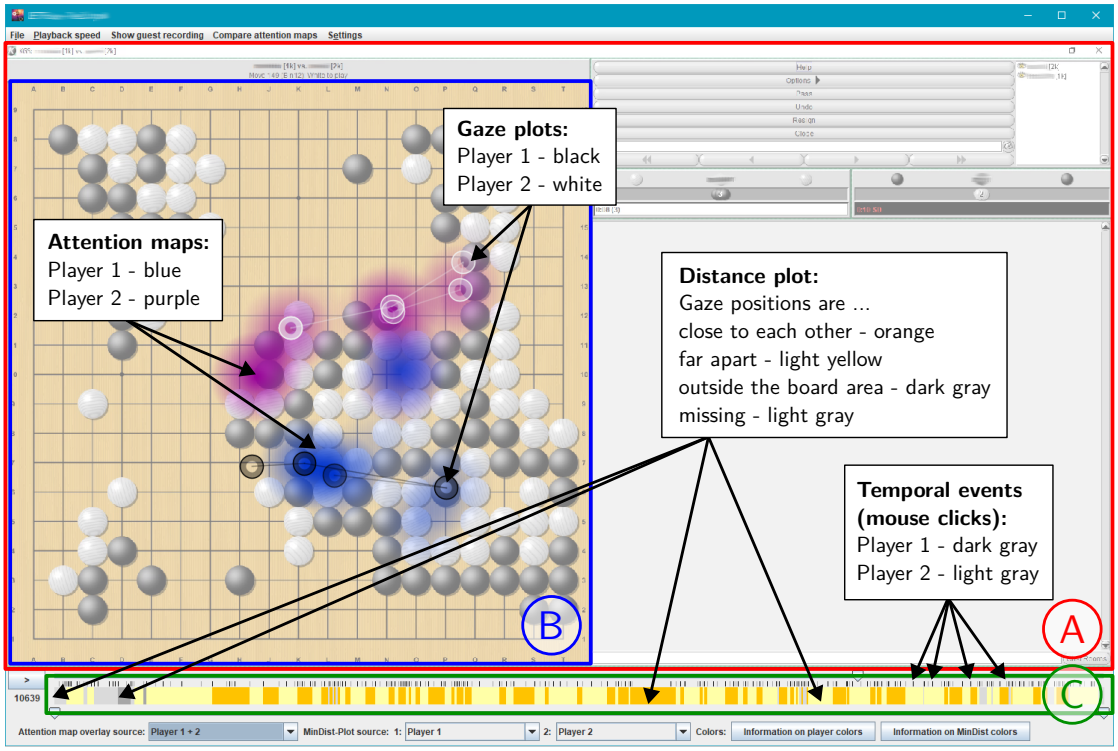


Figure 3.11 — Interface of our visual analysis system representing the eye movement data of two people playing the board game Go. The main area (A) shows the screen content of the first player (black stones), which contains the board (B) and a control panel to the right. Overlaid on the board are attention maps (blue and purple hotspots) and gaze plots (black and white circles connected by line segments) for a specific time frame for both players. (C) Below the main area, we provide a timeline highlighting important events as well as a distance plot that visualizes the proximity between the players' gazes.

This approach also deals with RQ1 and RQ2 similar to the system presented in the previous section: an interactive tool to explore the temporal eye tracking data with multiple visualizations is presented. Both static visualizations and animations can be used to explore the data. Additionally, parameter choices for the joining of the data from different devices influence how well the data streams are aligned and how well the comparison of the data can be performed. Further, different interval sizes can be selected for aggregated visualizations (attention map and distance plot) that may also influence what is perceived from the data during analysis.

The main contribution of this work is to provide a synchronization and combined visualization approach for eye movement data collected on independent and possibly different hardware for multiple participants. Hereby, different

methods to merge the data are presented. The source code, an example dataset, and a video for demonstration are publicly available [11]. This work is an extension of the bachelor's thesis of Noel Schäfer [279].

3.2.1 Related Work

We discuss the two research areas related to our work: eye tracking in the context of games and visualization of eye movement data.

Eye tracking for games [38] has become increasingly important in recent years, as discussed, for example, in EyePlay [316]. Related work can be separated into research on live systems communicating gaze positions between people and research on post-study analysis [73].

Smith and Graham [292] investigate gaze as an input modality for video games but do not consider gaze as a visual cue for other players. Sundstedt [303] gives a general overview of gaze input in games. Isokoski et al. [170] provide multiple examples of gaze-controlled games, including the board game Go. Lankes et al. [199] and Newn et al. [232, 233] provide gaze visualizations to players of a competitive board game and analyze how this cue influences their strategies. Maurer et al. [220] conduct similar research on collaborative games and Niehorster et al. [235] on collaborative and competitive visual search tasks. Furthermore, Vertegaal [325] presents a live system for gaze-aware communication between multiple people. Related work focuses on including gaze into remote scenarios, for example, to improve avatar communication or collaboration [99] and teaching [296, 346] scenarios. Further, Špakov et al. [295] use a VR context and share the focus between two players. The analysis of recorded gaze for post-hoc analysis is not the focal point of the publications mentioned above. Our work does not focus on gaze-aware communication but aims to provide detailed spatial and temporal analysis methods to investigate gaming scenarios with two players. However, because spatial and temporal matching between data sources is performed automatically, we see the potential of our approach also to be applied for live visualization of gaze, for example, in training for complex games, similar to EyeChess [293].

Gaze is a valuable component for game analytics [110], investigating the players' strategies and behavior [304]. Choi and Kim [87] and Almeida et al. [37] investigate the gaze distributions in first-person shooter games. Jermann et al. [174] use dual eye tracking and apply descriptive statistics to the gaze of two people playing a collaborative game. Shvarts et al. [287] synchronize and analyze data collected from head-mounted eye tracking devices. Charness et al. [82] compare expert and intermediate chess players' strategies but do not investigate the dynamics between players' gazes during a game. Kumar et al. [190] perform

a visual analysis of checker games, focusing on analyzing gaze distributions. Hessels et al. [150, 151] analyze the gaze of people looking at each other. In contrast, we introduce an approach for spatio-temporal analysis with specific comparison methods that help effectively analyze two players together.

Visualization plays a vital role in assessing statistical results and data exploration for hypothesis building. A survey of different techniques for eye tracking visualization is provided by Blascheck et al. [60]. Based on a taxonomy of eye tracking analysis tasks [195], techniques that enable comparing players viewing the same stimulus include timelines, attention maps, or gaze plots; also see Section 2.4.4 for details on these visualization types.

Timeline visualizations show the temporal evolution of gaze information. For example, gaze stripes [194] display a portion of the stimulus around a fixation as a timeline. However, we cannot use this technique because the stimulus snippets would be too similar due to the regular shape of a board game. As we are interested in the temporal similarity and difference of the participants' focus, we use a comparative timeline visualization. Additionally, we apply attention maps independently for both participants and show them on top of each other as an overlay on the stimulus to enable the exploration of differences and similarities. Gaze plots show the spatio-temporal order of fixations and saccades on a stimulus. This leads to visual clutter if gaze plots of all participants are depicted together. However, if only selected participants are compared or only short time spans are analyzed, in addition to animated changes [337], this technique enables analysts to compare eye movement data of participants on animated stimuli. This technique is helpful in our approach, to explore and compare the reaction to sudden changes in the stimulus.

3.2.2 Requirements

Our goal is to provide a way to analyze and compare eye movement data from two players who play collaborative and competitive board games against each other. The dynamic stimuli show the same gameplay to both participants, albeit on different computers that the participants are using to play against each other. This means that each player sees—at least for the most part—the same screen content, and for each player, the eye movements are recorded independently with stationary eye tracking systems. The input data required for our approach are each participant's gaze positions, screen recordings, and temporally logged events (e.g., mouse clicks). Such data might be obtained from recording devices with different hardware (screen resolution, sampling rate of the eye tracking systems), and the recordings might have started at different times. Therefore,

the data has to be temporally synchronized and mapped to a common screen area for the analysis. The following aspects have to be considered:

A1 Bi-directional mapping of gaze to the screen recordings.

A2 Temporal synchronization of data.

A3 Handling different recording frequencies.

After fusing the data, we support visual analysis with appropriate visualizations and interactions to explore interesting areas of the recordings. As a team of visualization researchers, eye tracking experts, and an experienced Go player (2-dan amateur player) who supported us in the analysis of our collected data, we identified the following requirements to compare eye movement data of multiple participants and support the visual analysis. More details about how we determined the requirements are available in Section 2.6.

R1 Subdivide time into intervals defined by important events that can be extracted from logged data (e.g., when a player performs a mouse click to put a stone on the board).

R2 Mark additional temporal positions (e.g., of interest to the analyst) and use them for the subdivision into intervals.

R3 Identify temporal areas for the analysis (e.g., during which gaze positions of multiple participants are close together).

R4 Allow visual comparison of gaze positions of multiple participants (e.g., with gaze plots and attention maps).

3.2.3 Visual Analysis Approach

With our visual analysis approach ETFuse (Figure 3.11), the comparison of eye movement data of two participants is possible. In the following, we outline our automatic approach to prepare eye movement data for analysis (A1–A3), and our visualization concepts for visual analysis (R1–R4) to detect and analyze interesting temporal and spatial areas within the recordings. Figure 3.12 shows an overview of our approach. Our system is implemented in Java.

Image-based Mapping and Synchronization

As the first step, the eye movement data from the two players has to be spatially registered (A1) and temporally synchronized (A2). Both can be achieved automatically by exploiting the image properties of the recorded videos.

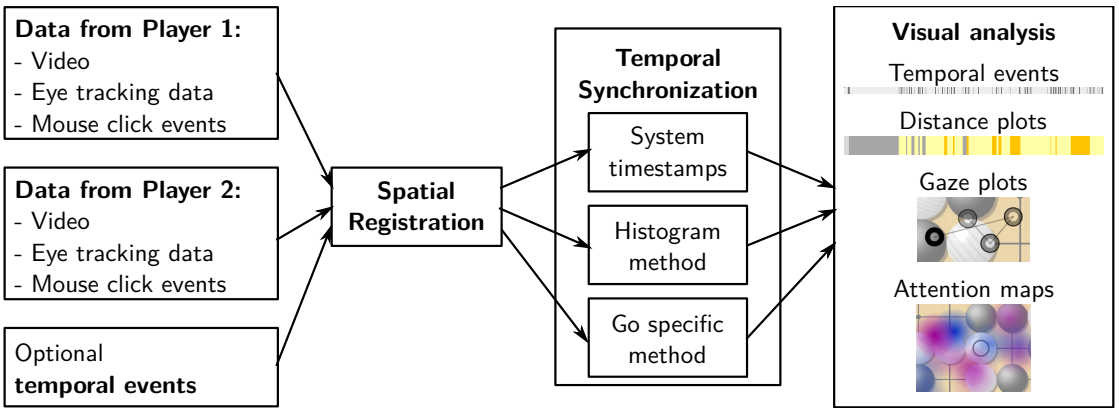


Figure 3.12 — Overview of our visual analysis approach.

Spatial Registration For the spatial mapping (A1), a coordinate transformation of the eye movement data of one of the players is performed to fit the recording for the other player. We model this mapping as an affine transformation and restrict it to the crucial part of the application for the analysis: the game board. The boards may be shown in different sizes for both players at different screen positions with different resolutions and aspect ratios. The remaining area on the screen may show different content for both players, such as additional graphical interface elements (e.g., a chat area) that are ignored from the mapping. In the analysis, users have to keep in mind that only the video of one player is shown. If eye movement data from the second player is displayed outside the specified region, this player might have seen a different screen content than the current video suggests. The analysis could be limited to the relevant cutout; however, it is still interesting to differentiate if a player looked elsewhere on the screen or if eye movement data was not available at all.

Temporal Synchronization We developed three methods to temporally synchronize (A2) the videos and eye movement data of two players. We assume that each player's video and eye movement data are synchronized.

A simple approach is using *system timestamps*. The synchronization accuracy depends on the quality of the system time on the individual machines used with the eye tracking software. In our experiments, we observed time differences between 0.3 and 5 seconds. It would be possible to set up additional time synchronization between the two machines to achieve higher accuracy. However, this would make our approach less flexible.

Our image-based *histogram method* can provide a more accurate result. Here, the goal is to identify the first frame in both videos that matches a given board configuration. Histograms are calculated for the individual color channels

and used to compare images for their similarity [79]. Multiple methods are available to compare histograms [80]; we use the Pearson correlation coefficient as a measure [240]. Our approach tries to find the first frame of a new board configuration (i.e., after putting a new stone on the board) by comparing histograms for the area of the board. With increasing screen resolution, the reliability of histogram comparison decreases as the number of images with similar histograms but different content increases. To avoid this problem, the frames of each video are divided by a grid into multiple cells to calculate and compare multiple histograms for each frame. For the selection of the grid size and the thresholds to detect similarity, it has to be considered that mouse cursors and half-transparent preview stones may influence the comparison and create wrong results. The first frame of a new configuration is found by repeatedly comparing it with previous frames until the histogram dissimilarity is above a certain threshold. Once this starting frame is found in the first video, we compare its histogram with frames from the second video until the earliest most similar frame is found. This approach also supports games other than Go and even other stimuli other than games.

We developed a third method that explicitly uses the context of a board game to find the frame when a specific stone was placed on the board in both recordings. The color at the position of the stone is compared for frames of the first video to detect when the stone was put on the board. Next, our method looks for the same color change for this stone in the other video and uses the time difference between both videos for synchronization. As before, mouse cursors and preview stones may influence the result. To avoid wrong synchronization that may be introduced by a color change of the mouse cursor, our method checks the color on two different positions on a stone, and the detection mechanism is based on a minimum color change. While this approach was designed for Go, it can also be used for other board games (e.g., chess) in which a sudden change of color for a specific position can be used for synchronization. This approach creates the most accurate results but may occasionally fail due to some different screen content (e.g., mouse cursor, preview stones, different textures of stones/board) shown on different machines.

The latter two methods may use the system timestamps to increase performance by providing estimated timestamps to find initial frames in the second video. This also avoids wrong synchronization results if the same board configuration is reached multiple times. Due to some varying time delays in the appearance of new elements resulting from the network connection, it is impossible to match the videos of the two players exactly. We observed a typical delay for the appearance of new elements in the range between 0 and 0.2 seconds after

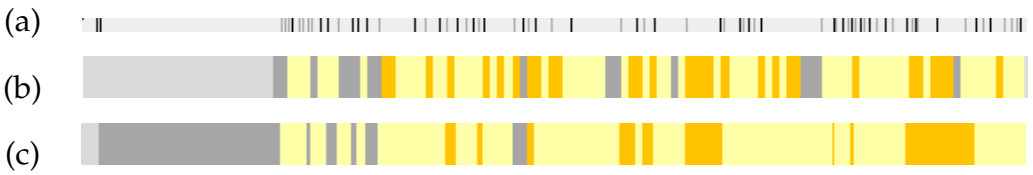


Figure 3.13 — (a) Timeline with positions of mouse clicks (lighter color for one player and a darker color for the other one). (b) Distance plot temporally subdivided by equal intervals of two seconds, and (c) by mouse clicks of both players.

the synchronization; this delay may occasionally be larger due to the network connection.

Visual Comparison Methods

Our visual analysis approach combines several visualizations for the analysis of collaborative and competitive content in a graphical user interface (Figure 3.11): a timeline highlighting important events (R1, R2; Figure 3.11 C), a distance plot (R3; Figure 3.11 C), gaze plots (R4; Figure 3.11 B), and attention maps (R4; Figure 3.11 B). For temporal subdivision, mouse clicks (R1) and analyst-defined additional temporal events (R2) are taken into account.

Mouse Clicks and Custom Events In a timeline, we highlight the position of important temporal events (R1, R2). The placement of a new stone is often connected with a change of attention for the players. A player triggers these events with a mouse click. The eye tracking system logs such events, and it is possible to visualize their temporal position (see Figure 3.13 (a)). Further events can be added manually to show them in the analysis. This is especially useful if interesting things happen on the screen that the two players do not influence. An example is when the players play against further opponents whose data is not recorded. Such events are shown in custom colors and can be used to subdivide the timeline into intervals (R2) for the distance plot and for updating the attention maps (see following paragraphs).

Distance Plots Below the timeline for events, we use a 1D plot to visualize the distance between the gaze positions of two players (R3). This visualization helps detect periods when players look at positions on the board that are close to each other. First, we divide time into intervals. This can be done for predefined equally sized time periods or according to specific events (see Figure 3.13 (b) and (c)). Then, we assign the color of one of four categories to each interval. Light gray is used when no eye movement data is available within the interval

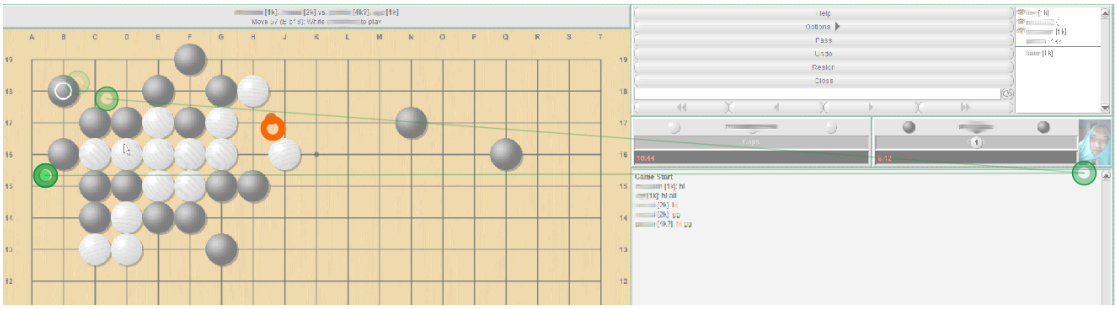


Figure 3.14 — Rengo game. Both players are on the same team, playing the white stones. Player 2 (orange fixation) looks intensely at one part of the board. Player 1, while also checking the board, glanced at the picture of a player on the opposing team (green circle on the right side).

for at least one player. This is often visible at the beginning or end of the timeline because the players usually start and stop their recordings at different times. Additionally, it may indicate that players looked away, closed their eyes, or, if intervals are small, that they blinked. Dark gray is used if most of the gaze positions are outside the board. Light yellow indicates that the distances between the gaze positions of both players on the board were mainly above a threshold. In all visualizations showing this approach, we use a threshold of 150 pixels, which equals about three cells on a board with a 19×19 grid. Orange highlights periods during which the distance was mainly below this threshold. As the frequencies of both recordings may differ, it is impossible to compare gaze positions directly. Instead, we upsample the lower-frequency data to then compare it to the gaze positions of the data recorded at the higher frequency (A3).

Gaze Plots A gaze plot (see Section 2.4.4) is a common visualization method used in eye movement research. It is often used for eye tracking data on static images but also used dynamically for video material (R4). As our stimulus changes over time, we show new gaze information in real-time. To keep track of the previous fixations, some of them are visualized as well. In our approach, we show joined gaze plots of multiple players (in different colors), in which all fixations are drawn by circles connected by straight lines representing saccades (see Figure 3.14). The most recent fixation is more opaque than the previous ones.

Attention Maps Attention maps (see Section 2.4.4) are a more scalable visualization for many gaze positions. They highlight areas that people focus on most. Our approach generates the attention maps for fixations within a specified

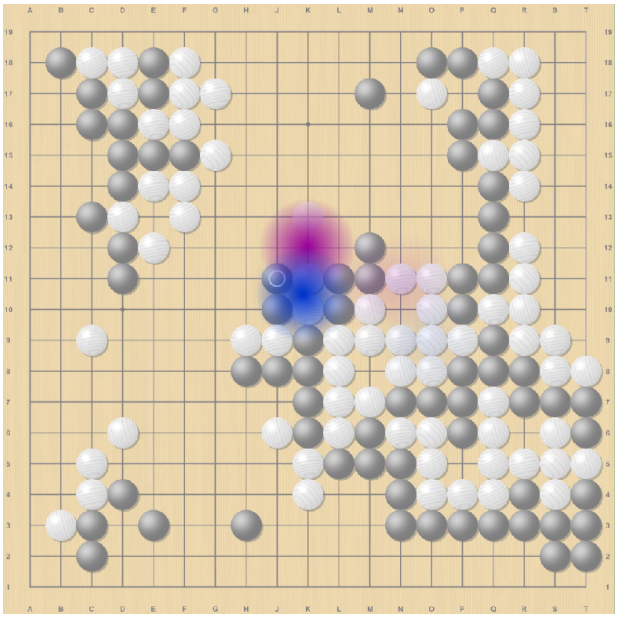


Figure 3.15 — Player 1 (black) is focusing on the two white stones in the center of the board (blue hotspot) and is ready to capture them, which would end the game. Player 2 focuses on the *only* escape route for the two white stones.

interval. It is possible to show the attention map of only one player or multiple players as overlays (R4). To display multiple attention maps simultaneously, each player is represented by a distinguishable color whose alpha channel goes from opaque to transparent. Figure 3.15 shows an example in which blue is used for one player and purple for the other player. The time interval for calculating attention maps can be an analyst-defined interval (e.g., for the entire time of a match) or multiple predetermined intervals to show dynamic changes in attention. The previously mentioned logged and manually specified temporal events can be used to calculate attention maps between every two events and to update the overlay according to the current temporal position.

3.2.4 Evaluation

To illustrate the effectiveness of our approach, we apply it to Go, the oldest known board game in human history whose rules have not changed since its inception. Go has high state-space and game-tree complexities [339] and has been recently solved (i.e., AI programs defeating the strongest human players) by *AlphaGo*, a software developed by Deep Mind based on DL [288, 289].

The standard Go board contains a 19×19 grid, on which the two players take turns to place a stone of their respective color (black or white). Stones already placed on the board by one player can be *captured* by the other player under certain conditions. At the end of the game, both players count their respective *territory*. The player with more points wins the game. In addition, Go can be played between two teams of players, called *Rengo* [286]. Typically, each team

consists of two players who take turns to place stones of their color on the board.

In the following, we use the results from some game scenarios as an illustration of the capability of our approach and to demonstrate how our approach can be used to explore the behavior of participants playing board games. Future work with a more extensive user study involving a larger number of participants would be required to strengthen our findings and discover new hypotheses.

Some possible questions that can be explored:

Q1 Is there a difference in focused areas between players, reflecting their intentions, experience, or styles?

Q2 Do players know what they want to do next or do they have to think about multiple possible next moves?

Q3 Is a player focused while it is the other player's turn?

Data Acquisition

We collected data from several Go matches during which two volunteers played the virtual board game KGS Go [125] via internet connection against each other or in collaboration against two other opponents whose eye movement data was not recorded. The first player has a strength of 1-kyu, which puts him at the top of the bracket of *intermediate amateurs*, and the second player has a strength of 2-dan, which puts him in the bracket of *advanced amateurs*. We used two different remote eye tracking devices to record the data, which differ in their recording frequency and monitor resolution (see page 26 in Section 2.4.1 for details on this type of eye tracking system). The first device was a Tobii Pro Spectrum 1200; it has a screen resolution of 1920×1080 pixels, eye movement data was recorded at 1200 Hz, and the Tobii Pro Lab software was used to record the eye movements and the screen. The second device was a Tobii T60XL; it has a screen resolution of 1920×1200 pixels, recorded at 60 Hz, and Tobii Studio collected all data.

The recordings were performed in a lab space isolated from outside distractions. Calibration and recording were run independently for both participants on their devices, and the players started a match once everything was set up. The participants played different types of matches against each other, leading to a variety of different game scenarios for the analysis. Each match lasted between 4 and 11 minutes, and matches were performed on different grid sizes.

The more experienced player (in the following Player 2) always played the white stones, whereas the other (Player 1) played the black ones. As it is common

in Go to start with black stones, Player 1 had the first move in each match. In Rengo, both players used white stones. When the maps for both players are shown simultaneously, the color of an attention map is blue for Player 1 and purple for Player 2. In Rengo, Player 1 has green fixations, whereas Player 2 has orange ones.

Visual Analysis

In this section, we demonstrate the effectiveness of our approach. One of the co-authors, a 2-dan amateur player (advanced amateur), performed the data analysis. Additionally, we showed our observations to the players and collected their feedback. We describe the findings from a Rengo game and one of the competitive matches; a 19×19 grid was used for both games.

When exploring the eye movement data, we could detect quite different playing styles [168] for both players (Q1) and observe their reading process during games [172]. In several games, we consistently observed an attack-oriented style for Player 1 and a defense-oriented style for Player 2. Both their thinking process and intentions differed in such scenarios, which might also be influenced by their strengths in the game. After we informed the players about our findings to their playing styles, they both said that it made sense. However, they were surprised that the eye movement data was able to capture this information. An example is visible in Figure 3.16 (left). During the Rengo game, the black team attacked the white stone in the lower left corner. Both players agreed that a local response was needed but with different strategies (Q1). Player 1 (green fixations) was in turn and, considered attacking the black stone by pincering it. Player 2 focused more on a position opposite the black stone to support the white stone (Q3). This difference in attention would not be that easily visible without visualization of the synchronized data.

Figure 3.16 (right) shows another situation from the Rengo game during which the players had different intentions (Q1). Right after the white team had captured a key black stone in the upper left, the two participants considered leveraging their newly gained strength to attack nearby black groups. Player 1 (green fixations) explored capturing the black group in the corner, while Player 2 (orange fixations) was considering attacking the black group on the outside, for which more points could be made.

Another example is visible in Figure 3.15; it shows the joint attention map during a game where the two participants played each other. The game was at a crucial stage, with two white stones being the center of a fight. Both players have their attention around the same place. However, Player 1 (black) looks at the two white stones (blue hotspot) and is ready to capture them and end the

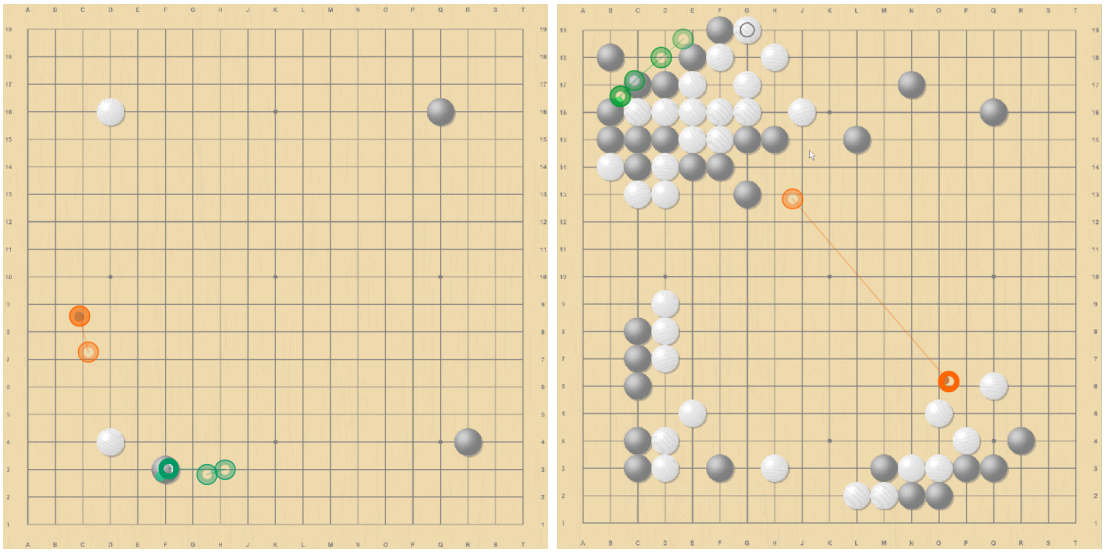


Figure 3.16 — Rengo game. Left: After the opposing team approached the white stone at the lower-left corner, the data on the eye movement of the two participants showed their different responses. Right: After strengthening their group that contained the stone with an inscribed black circle, both participants planned an attack. Their gaze reveals that they were focusing on different targets.

game. However, it was the turn of Player 2, whose attention (purple hotspot) was fixed on the grid above, which was the only escape route for his two stones. It is visible that Player 1 also thinks about a possible next move as if it were his turn and not about a possible response of Player 2 (who actually is in turn) to his previous move (Q2, Q3).

Additionally, we noticed in most matches that the attention maps for the entire match (or larger time spans) showed that Player 1 looked at broader areas of the board while Player 2 focused more on specific regions (Q1, Q2). For example, in the competitive matches Player 2 mainly focused on the bottom right part of the board. While Player 1 also focused on this region, his gaze was generally more spread and toward the top of the board. In the Rengo game, the difference between this focus was also visible. Especially at the beginning of the match, both participants focused on different areas, and only later the areas they focused on became quite similar.

Exploring the gaze plots of the players revealed further patterns in the focus of the players (Q1, Q3). For example, at one point during the Rengo game (Figure 3.14), it was the turn of Player 2. While he was carefully considering how to proceed with the local fight (focused gaze indicated by the sole orange

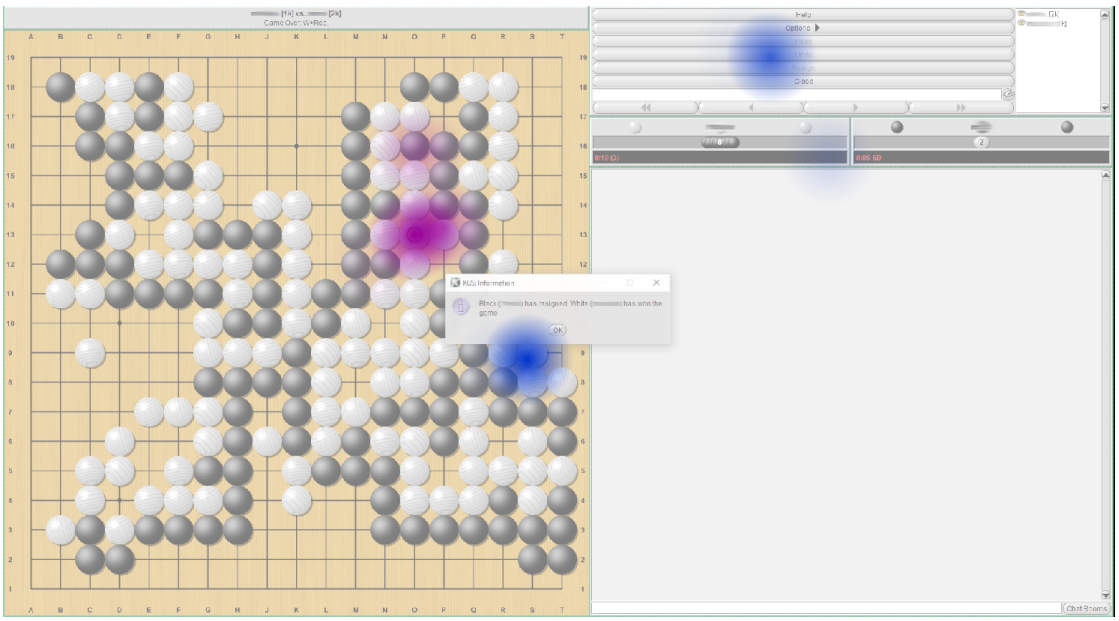


Figure 3.17 — Player 2 (white) played the stone (purple hotspot) to win the game and his focus remains on this position on the board. Player 1 looks at the resign button (blue hotspot on the top) and the dialog that appears after pressing the button (blue hotspot in the center).

circle in the middle of the board), Player 1 (on the same team but out of turn) was also examining the local situation. However, his gazes at the upper-left corner of the board sandwiched a quick glance at one of the opposing player's icons on the right side of the interface (green fixations). This quick attention change was probably due to two factors: Player 1 was not in turn, so there was less pressure on him. Player 1 laughed when we showed him that we recorded him looking at the opponent's icon. He said, he was curious about people's KGS names and icons. If they look interesting, he would consider playing more games with them. With our approach, it is easier to make such observations because it is possible to study the eye movements of both players simultaneously and not with two separate recording streams.

We made another interesting observation in the same game (Figure 3.17). Both players drew the same conclusion from the current situation but with a different behavior (Q1). Player 2 placed a stone (below the purple hotspot) to get his stones back to safety and win the game. Afterward, his eye movement stayed on this spot, and he looked nowhere else because the game was over. At the same time, Player 1 (blue hotspot) looked at the resign button and the dialog that appeared after pressing the button. Despite the relatively large distance

between the areas of attention of both players, their conclusions regarding the game were the same.

Finally, we noticed that both players have the habit of controlling their own and the opponents' remaining time by looking at the control panel; this may look like they are not paying attention.

General Observations

In the analysis, we mainly focused on gaze plots and attention maps. It was easier to keep track of them as they were shown on the board, and it was possible to monitor the game situation simultaneously.

We experienced that gaze plots are more effective than attention maps for this use case because the former shows the exact grid points that have been focused on and the sequence in which these grid points were gazed at. For example, in Figure 3.14 on page 87, the gazes of Player 1 shifted from the board to an opponent's icon and back. The attention map can show the two areas the player focused on but does not reveal the back-and-forth shift in his focus.

In contrast, the attention maps appear to be more stable as they were time-averaged. Moreover, they can provide a better summary for multiple gaze positions than the gaze plots. For example, in Figure 3.18, the attention of Player 1 had spread over six different parts of the board and interface, including checking the remaining time for his team. Such information would be difficult to find from the gaze plots that we are using to see the current eye movement of both participants, and the number of fixations is thus limited.

When studying the attention maps for the entire duration of a match, we noticed for one of the 19×19 competitive matches that attention maps cover grid points that were eventually occupied. This shows that places the players considered as a possible next move eventually became the next move (Q2). Another observation is that places where heavy local fights occurred, tend to be covered by the attention maps. As mentioned before, we also detected differences between the players and the focused areas (Q1). We hypothesize that professional players could use such maps to spot blind spots of amateur players.

The distance plots were also helpful in the analysis. At the beginning of the game, when the board is relatively open, the players' choices of a next move are largely dependent on their level and style (Q1); this is when their attention diverges the most (see Figure 3.13 on page 86: first, the gray areas indicate that the players are not yet ready; then, light yellow areas dominate as players look at different board areas). However, in the middle and end of games, players are engaged in local fights that attract their attention to the same parts of the

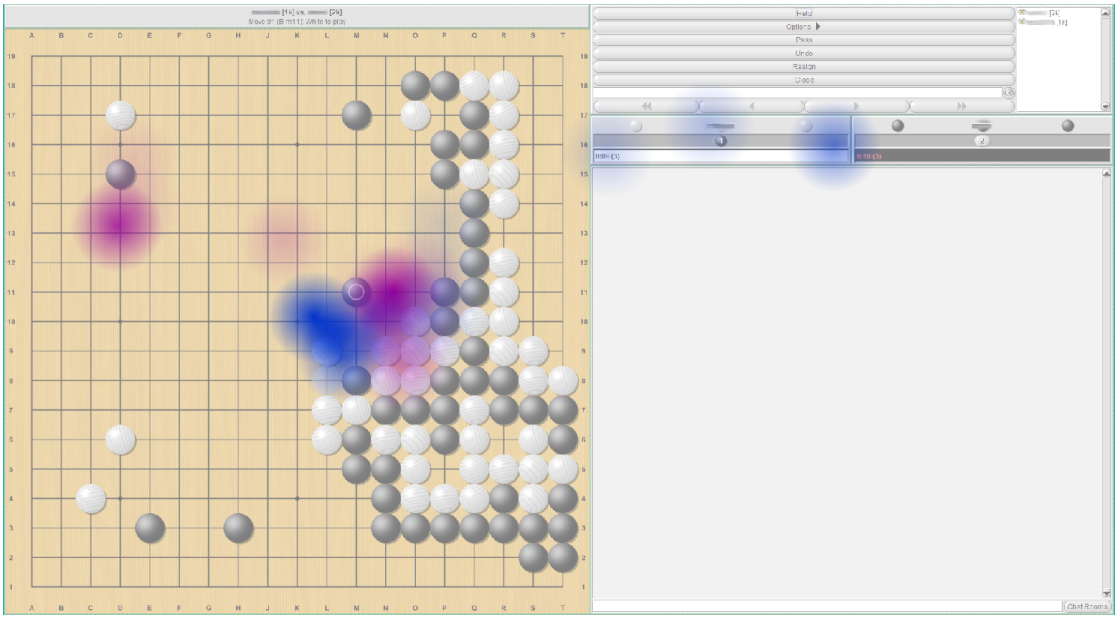


Figure 3.18 — After the two players were engaged in a *semeai* (Go term for a capture race), the fight resulted in a *seki* (both groups live). Both players recognized this outcome and immediately shifted their focus to other parts of the board.

board. In such cases, we can see areas in the distance plot that indicate that the gaze positions of both players are close together (see Figure 3.13: larger orange areas show positions of common attention). Of course, when a local fight is settled, the players are open to deciding where the next move is; thus, styles set in again with diverging gazes (Q1, light yellow). We believe this feature could be used in more extensive studies to quickly find areas of local fights or areas when players are distracted by the chat area or not looking at the screen at all and to explore the duration of such events.

It was essential for the analysis to see when mouse clicks were used to place new stones. The event timeline can also provide a sense of how quickly a player responds when it becomes his turn. For instance, players might think more when the game is in a complicated situation. Overall, Player 1 spends less time between moves than Player 2.

When we showed the recordings to the players, they liked the attention maps and gaze plots the best. After understanding the distance plots, they were interested in the Rengo game and wanted to know how much they agreed with each other. Player 2 also wanted to find out where and why they disagreed and hypothesized that they were different because he was stronger.

Some of the patterns we found can also be seen from the placement of stones. However, if we analyzed only newly placed stones without gaze information, it would not be possible to observe whether players explored the same or different regions of the board, thought about their own next moves or the ones of the opponent, explored a broader area of the board or focused on small regions and would play alike or differently, if it was the other player's turn (Q1–Q3). Especially in collaborative games, we see a high value of the additional information from eye movements as only the moves of one player are visible by the appearance of new stones and not what the intention of the other player might have been (Q1, Q3).

3.2.5 Conclusion and Future Work

We presented ETFuse, a new approach for the combined visual analysis of eye movement data from two players in virtual board games. Data recorded with two independent eye tracking systems can be automatically mapped to the same board area and temporally synchronized to compare both players visually. The mapped and synchronized data can be visually analyzed based on mouse clicks and custom events using a distance plot, which provides a temporal summary of the distance between players' gaze positions, attention maps showing the overall or time-constrained attention, and gaze plots of specific time periods for both players. Our use case demonstrated that different strategies of the two players could be extracted. Such intense and contrasted focus by two players would be more difficult to observe without using a synchronized visualization approach.

To this point, we focused on analyzing two players with a common board to which we map the gaze of both players. An extension for multiple players and more complex competitive multi-player games such as real-time strategy or 3D games would be possible. While we used data collected from two amateur players, recordings from more players with different experiences would allow an analyst to compare the players' strategies for different games.

3.3 Summary and Conclusion

This chapter presented two visual analysis approaches for temporal data from eye tracking. The first approach focuses on the possibility of changing parameter settings during the interactive filtering for microsaccades. Different visualizations for analyzing eye movements, especially microsaccades for one or multiple people, are provided. The second approach, instead, focuses on the analysis of the eye movements of two people playing an online board game. For a comparison of their eye movements, the data originating from two different recording setups must be merged. Given the possibility of joining data from different hardware, our approach is very flexible in its usage.

The approaches presented here deal with RQ1 and RQ2: the visual analysis of sequential data and the influence of parameter settings on the analysis. Both presented approaches benefit from the flexible use of different visualizations (RQ1) that makes it easy to focus on specific parts of the data and support a comparison between different temporal areas. Additionally, for filtering eye movements, parameters have to be defined for the algorithm that detects them (RQ2). Only well-chosen values detect eye movements appropriately. In our research, we noticed that different research papers use different values and algorithms. This makes it challenging to compare analysis results. With our visual analysis system, we explored the strong influence of these values on the number, size, and quality of detected movements. If the movements are not correctly detected, this affects the whole analysis, including the visualizations presented and the conclusions that can be drawn. The same applies if the synchronization of the two eye tracking recordings cannot be performed well. Additionally, parameter settings have, for example, an impact on the visualizations when deciding the size of bins or intervals when aggregating or averaging some data. For example, if the bin sizes are too large or too small, important information may be missed or larger patterns might not be visible. This could be observed in both approaches.

CHAPTER 4

Visual Analysis of Deep Learning with Sequential Aspects

The previous chapter focused on the visual analysis of temporal eye tracking data. This chapter addresses DL models where sequential aspects are crucial. Two visual analysis approaches are presented that contribute to the research area of XAI (Section 2.5.3). Both approaches are concerned with the issue of better understanding NLP prediction models. In the presented techniques, text sequences are used as input to ML models. The goal of the visual analysis is to increase the interpretability of the underlying ML mechanisms and to improve the performance of the predictions. While the first approach of this chapter (Section 4.1) focuses on NMT, the second one (Section 4.2) deals with VQA. Information on these two DL approaches can be found in Section 2.5.2. For both approaches, the input data and prediction mechanisms depend on sequences and sequential processing of information. A further approach of this thesis with similar principles and goals (Section 5.1) would also fit here. However, since it has an additional focus on using dimensionality reduction for visualization, it was instead placed in the next chapter.

4.1 Neural Machine Translation

In this section, a visual analytics approach for ML-based document translation (Neural Machine Translation Visualization System (NMTVis)) [15, 19] is pre-

sented. It helps analyze, understand, and correct results created with NMT with the help of multiple interactive visualizations.

This section is based on the following publications:

- T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. Visualization-based improvement of neural machine translation. *Computers & Graphics*, 103:45–60, 2022, doi: 10.1016/j.cag.2021.12.003 [19].
- T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. Visual-interactive neural machine translation. In *Proceedings of Graphics Interface 2021 (GI 2021)*, pages 265 – 274. Canadian Information Processing Society, 2021, doi: 10.20380/GI2021.30 [15].

The images shown in this section were created with the following source code/material:

- T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. NMTVis - Extended neural machine translation visualization system. DaRUS, V1, 2022, doi: 10.18419/darus-2124 [18]. (The source code is also available on GitHub: https://github.com/MunzT/NMTVis.)
- T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. NMTVis - Trained models for our visual analytics system. DaRUS, V1, 2021, doi: 10.18419/darus-1850 [17].

Machine translation models (see page 40 in Section 2.5.2) are computationally efficient and able to translate large documents with low time effort, but they may create erroneous or inappropriate translations. Humans are very slow compared to these models, but they can detect and correct mistranslations when familiar with the languages and the domain terminology. In a visual analytics system, both of these capabilities can be combined if high-quality translations for large texts are required. Our system (Figure 4.1, Figure 4.2) performs automatic translation of a whole, possibly large, document and shows the result in the document view. Users can then explore and modify the document in different views [273] to explore and improve translations and use the corrections to fine-tune the NMT model. We support different NMT architectures and use both an LSTM-based (page 45 in Section 2.5.2) and a Transformer (page 47 in Section 2.5.2) architecture. In this approach, the sequential component is present as input and output data (sentences and their translation) and also during the prediction of a new translation. Here, especially the attention and the beam search views (see page 112 in Section 4.1.3) visualize sequential components. These visualizations are strongly linked to the underlying model: visualized attention weights correlate with the importance of source words for the translated words, and multiple translation possibilities created by the beam search decoding are presented to the user.

This approach contributes to RQ1, RQ3, and slightly to RQ2: Multiple interactive visualizations are provided to explore textual sequences in the context of

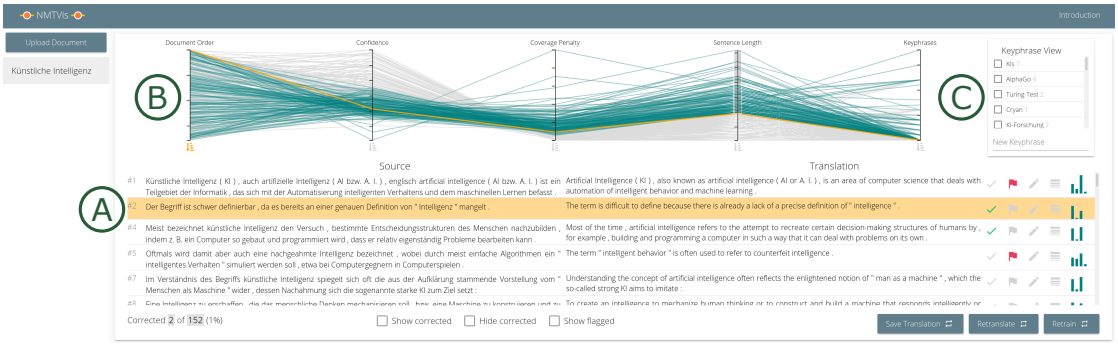


Figure 4.1 — The main view of our NMT system: (A) Document view with sentences of the document for the current filtering settings, (B) metrics view with sentences of the filtering result highlighted, and (C) keyphrase view with a set of rare words that may be mistranslated. The document view initially contains all sentences automatically translated with the NMT model. After filtering with the metrics view and keyphrase view, a smaller selection of sentences is shown. Each entry in the document view provides information about metrics, the correction state, and functionality for modification (on the right side next to each sentence). The metrics view represents each sentence as one path and shows values for different metrics (e.g., correlation, coverage penalty, sentence length). Green paths correspond to sentences of the current filtering. One sentence is highlighted (yellow) in both the metrics and document view. This image was created for a document translated with a Transformer model.

Vis4ML. Both the beam search view and the attention view (graph and matrix-based; Figure 4.2 B and C) provide visualizations for sequences to gain a deeper understanding of the mechanisms of the underlying prediction model. Additionally, the beam search view (Figure 4.2 D) can be used interactively to explore and change the translation. These visualizations help explore sequences (RQ1) and internal behaviors of DL (RQ3). Additionally, this approach provides possibilities to change parameters for the visualized information (RQ2). An example is the number of alternative translations being shown in the beam search view. Depending on the selected beam size, a varying number of translations with different suitability are suggested.

This work contributes to visualization research by introducing the application domain of NMT using a user-oriented visual analytics approach. Our system employs different visualization techniques adapted for usage with NMT and is therefore derived from various existing approaches. We combined individual components of these approaches and adapted them to our needs; a concept

Figure 4.2 — The detailed view for a selected sentence consists of the sentence view (A), the graph-based attention view (B), the matrix-based attention view (C), and the beam search view (D). The sentence view allows text-based modifications of the translation. The attention views show the attention weights for the translation between source words and translated words. The graph-based visualization represents attention values by the lines connecting source words with their translation. The matrix-based visualizations show the corresponding values by coloring the corresponding grid cells between source words and their translation. The beam search view provides an interactive visualization that shows different translation possibilities and allows exploration and correction of the translation. All four areas are linked. On the bottom, users can change some parameters for the visualizations and the models. This image was created from a sentence translated by an LSTM model.

mentioned in Section 2.3. We added new interaction techniques to some visualizations for NMT in this context. Our parallel coordinates plot (Figure 4.1 B)

supports the visualization of different metrics related to text quality. The interaction techniques in our graph- and matrix-based visualizations for attention (Figure 4.2 B and C) and tree-based visualization for beam search (Figure 4.2 D) are specifically designed for text exploration and modification. They have a strong coupling to the underlying model. Additionally, we enable the use of subword units and differentiate between them and whole words in the beam search visualization and matrix-based attention view. Our workflow is not limited to only exploring and debugging NMT models with visualizations; it can be used for the whole translation process of documents. Even adapting the NMT model to the current vocabulary is possible. Our system is developed generically for NMT models, and we used it with an LSTM-based and a Transformer model. We publicly provide the source code [18] of our system and the trained models [17] we used in our evaluation. This work expands on the master's thesis of Paul Kuznecov [196].

4.1.1 Related Work

This section first discusses the visualization, visual analytics, and interaction approaches for language translation in general and then visual analytics of DL for text. Afterward, we provide an overview of work that combines both areas in the context of NMT.

Many visualization techniques and visual analytics systems exist for text; see Kucher and Kerren [189] for an overview. However, there is little work on exploring and modifying translation results. An interactive system to explore and correct translations was introduced by Albrecht et al. [34]. While the translation was created by machine translation, their system did not use DL. Lattice structures with uncertainty visualization were employed by Collins et al. [92] in the context of machine translation. They created a lattice structure from beam search where the path for the best translation result is highlighted and can be corrected. We also use visualization for beam search, but ours is based on a tree structure. Without the use of visualization, Green et al. [136] follow a similar approach to ours to let users correct machine-translated sentences providing suggestions. They discuss that post-editing of mistranslated sentences reduces time and creates results with better quality [136, 137].

Recently, much research was done to visualize DL models to understand them better (see Section 2.5.3). It is noticeable that not much work exists related to text-based domains. One of the examples is RNNVis [228], a visual analytics system designed to understand and compare models for NLP by considering hidden state units. Karpathy et al. [178] explore the prediction of LSTM models by visualizing activations on text. Heatmaps are used by Hermann et al. [148]

in order to visualize attention for machine-reading tasks. To explore the training process and to better understand how the network is learning, RNNbow [78] can be used to visualize the gradient flow during backpropagation training in RNNs.

While the previous systems support the analysis of DL models for text domains in general, approaches exist to specifically explore and understand NMT. The first who introduced visualizations for attention were Bahdanau et al. [46]. Using an attention weight matrix, they showed the contribution of source words to translated words within a sentence. We also include an interactive version of such a visualization. Later, Rikters et al. [272] introduced multiple ways to visualize attention and implemented exploration of a whole document. They visualize attention weights with a matrix and a graph-based visualization connecting source words and translated words by lines whose thickness represents the attention weight. Bar charts give an overview of a whole document for multiple attention-based metrics that are supposed to correlate with the translation quality. Interactive ordering of these metrics and sentence selection is possible. However, it is difficult for large documents to compare the different metrics as each bar chart is horizontally too large to be entirely shown on a display. The only connection between different bar charts is that the bars for the currently selected sentence are highlighted. Our system also uses such a metrics approach, but instead of relying on bar charts, a parallel coordinates plot was chosen for better scalability, interaction, and filtering. Additionally, we performed a computer-based evaluation for our chosen metrics.

An interactive visualization approach for beam search is provided by Lee et al. [202]. The interaction techniques supported by their tree structure are quite limited. It is possible to expand the structure and change attention weights. However, it is not possible to add unknown words, and no subword units are considered. Furthermore, the exploration is limited to single sentences instead of a whole document. In our approach, we deal with these limitations.

With LSTMVis, Strobelt et al. [300] introduced a system to explore LSTM networks by showing hidden state dynamics. Among other application areas, their approach is also suitable for NMT. While our approach is intended for end-users, LSTMVis aims to debug models by researchers and ML developers. With Seq2Seq-Vis, Strobelt et al. [301] present a system that uses a graph-based attention view similar to ours and provides an interactive beam search visualization. However, their system is designed to translate single sentences, and no model adaption is possible for improved translation quality. Their system aims to debug and gain insight into the models.

Since different architectures are available for generating translations [344], specific visualization approaches may be required. Often, LSTM-based architectures are used. Another more recent architecture is the Transformer [322]. Vig [326, 327] visually explores its self-attention layers, and Rikters [271] extended their previous approach for debugging documents to Transformer-based systems.

All these systems provide different, possibly interactive, visualizations. However, their goal is to debug NMT models instead of supporting users in translating entire documents, or they are limited to small aspects of the model. Additionally, they are usually designed for one specific translation model. None of these approaches provide extended interaction techniques for beam search or interactive approaches to iteratively improve the translation quality of a whole document.

4.1.2 Requirements and Design

Our approach is strongly linked to machine data processing and follows the visual analytics process presented by Keim et al. [182]. We use visualizations for different aspects of NMT models, and users can interact with the provided information.

Requirements

We followed the nested model by Munzner [231] to develop our system. The main focus was on the outer parts of the model, including identifying domain issues, feature implementation design, and visualization and interaction implementation. Additionally, we used a similar process as Sedlmair et al. [281], mainly focusing on the core phases. Design decisions were made in close cooperation with DL and NMT experts, who are also co-authors of the publications related to this work [15, 19]. The visual analytics system was implemented in a formative process involving these experts. Our system went through an iterative development that included multiple meetings with our domain experts. Together, we identified the requirements listed in Table 4.1. After implementing the basic prototype of the system, we demonstrated it to further domain experts. At a later stage, we performed a small user study with experts for visualization and machine translation. For our current prototype, we added the functionality recommended by these experts. Refer to Section 2.6 for more details on how we derived the requirements.

Table 4.1 — Requirements for our visual analytics system and their implementations in our approach.

R1	**Automatic translation** – A document is translated automatically by an NMT model.
R2	**Overview** – The user can see the whole document as a list of all source sentences and their translations (Figure 4.1 A). Additionally, the metrics view provides an overview of the translation quality, revealing statistics about different metrics encoded as a parallel coordinates plot (Figure 4.1 B) showing an overall quality distribution.
R3	**Find, filter, and select relevant sentences** – Interaction in the parallel coordinates allows filtering according to different metrics and selecting specific sentences. It is also possible to select one sentence and order the other sentences of the document by similarity to verify for similar sentences if they contain similar errors. Additionally, our keyphrase view (Figure 4.1 C) supports selecting sentences containing specific keywords that might be domain-specific and rarely used in general documents.
R4	**Visualize and modify sentences** – For each sentence, a beam search and two attention visualizations (Figure 4.2) can be used to interactively explore and adapt the translation result in order to correct erroneous sentences and explore how a translation failed. It is also possible to explore alternative translations.
R5	**Update model and translation** – The model can be fine-tuned using the user inputs from translation corrections. This is especially useful for domain adaption. Afterward, the document is retranslated with the updated model to improve the translation result (Figure 4.5 on page 108).
R6	**Generalizability and extensibility** – While we initially designed our visualization system for one translation model, we soon noticed that our approach should handle data from other translation models as well. Therefore, our approach should be easily adaptable for new models to cope with the dynamic development of new DL architectures. Our general translation and correction process is quite agnostic to be applied to various models. Model-specific visualizations may have limitations and must be adapted or exchanged using a different translation architecture.
R7	**Target group** – The target group for our system should be quite broad and include professional translators or students who need to translate documents. However, it should also be able to be used by other people interested in correcting and possibly better understanding the results of automated translation.

Neural Machine Translation

Machine translation (see page 40 in Section 2.5.2) aims to translate a sequence of words from a source language into a sequence of words in a target language. Usually, NNs for machine translation are based on an encoder-decoder architecture (see page 46). The decoder generates an output sequence where each element is used to generate a probability distribution over the target vocabulary. These probabilities are then used to determine the target sequence. A common method to achieve this uses beam search decoding [135].

Although different NMT models vary in their architecture, the previously mentioned encoder-decoder design should apply to a wide range of architectures and new approaches that may be developed in the future (R6). In this work, we explored an LSTM architecture with attention and extended our approach to include the Transformer architecture, thus verifying its ability to generalize (see page 45 in Section 2.5.2 for details on these models).

The attention mechanism for NMT [46] was introduced to handle long sentences. It allows Seq2Seq models to pay attention to different sections of the input sequence while predicting the next item of the output sequence by providing the decoder access to the encoder's weighted hidden states. The attention weights can be easily visualized and used to explain why an NN model predicted a specific output. Furthermore, the attention weights can be seen as a soft alignment between source and target sequences. For each translated word, the weight distribution over the source sequence indicates which source words were most relevant for predicting that target word.

The Transformer architecture was introduced by Vaswani et al. [322]. It uses a more complex attention mechanism with multi-head attention layers. Self-attention plays a vital role in the translation process. We verify its applicability to our approach and visualize only the part of the attention information that showed an alignment between source and target sentences comparable to the LSTM model.

4.1.3 Visual Analytics Approach

Our visual analytics approach NMTVis allows the automatic translation, exploration, and correction of documents. Its components can be split into multiple parts: (1) A document is automatically translated from one language into another one. (2) Users identify mistranslated sentences in the document. (3) The users can explore and correct individual sentences. (4) The model can be fine-tuned, and the document can be retranslated. This workflow is also shown in Figure 4.3.

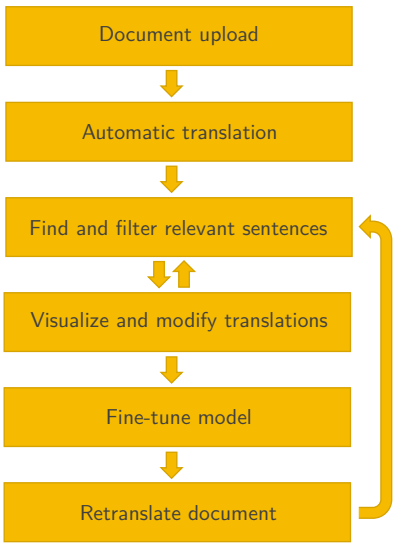


Figure 4.3 — Workflow of our system: First, a document is uploaded to our system and automatically translated. Next, sentences can be filtered and selected for visualization and modification. Finally, the model can be fine-tuned, and the document can be retranslated. The last steps can be repeated multiple times until the translation quality is sufficient.

Exploration of Documents

After uploading a document to our system, it is translated by an NMT model (R1). The main view of our approach then shows information about the whole document (R2). This includes a list of all sentences in the document view (Figure 4.1 A) and an overview of the translation quality in the metrics view (Figure 4.1 B). Using the metrics view and keyphrase view (Figure 4.1 C), sentences can be filtered to detect possible mistranslated sentences that can be flagged by the user (R3). Once a mistranslated sentence is found, it is also possible to filter for sentences containing similar errors (R3).

Metrics View In the *metrics view*, a parallel coordinates plot (Figure 4.1 B) is used to detect possible mistranslated sentences by filtering sentences according to different metrics (R3). For instance, it is possible to find sentences with low translation confidence.

Multiple metrics exist that are relevant to identify translations with low quality. We use the following metrics in our approach:

- **Confidence:** A metric that considers attention distribution for input and output tokens suggested by Rikters et al. [272]. Here, a higher value is usually better.
- **Coverage penalty:** This metric by Wu et al. [343] can be used to detect sentences where words did not get enough attention. Here, a lower value is usually better.
- **Sentence length:** The sentence length (the number of words in a source sentence) can be used to filter very short or long sentences. For example, long sentences might be more likely to contain errors.

- **Keyphrases:** This metric can be used to filter for sentences containing domain-specific words. As these words are rare in the training data, the initial translation of sentences containing them is likely erroneous. The values used for this metric are the number of occurrences of keyphrases in a sentence weighted by the frequency of the keyphrases in the whole document.
- **Sentence similarity:** Optionally, for a given sentence, the similarity to all other sentences can be determined using cosine similarity. This helps find sentences with similar errors to a detected mistranslated sentence.
- **Document index:** The document index allows the user to sort sentences according to their original order in the document, which can be especially important for correcting translations where the context of sentences is relevant. Furthermore, this metric might also show trends like consecutive sentences with low confidence.

In contrast to Rikters et al. [272], who use bar charts to visualize different metrics, we chose a parallel coordinates plot [166]. Each sentence can be mapped to one line in such a plot, and different metrics can be easily compared. Such a plot is useful for an overview of different metrics and detecting outliers and trends. Interactions with the metrics, such as highlighting lines or choosing filtering ranges, are supported. It can be expected that sentences filtered for both low confidence and high coverage penalty are more likely to be poorly translated than sentences falling into only one of these categories.

Keyphrase View It is possible to search for sentences according to keyphrases by selecting them in the *keyphrase view* (Figure 4.1 C) (R3). The results can be visualized as shown in Figure 4.4. Keyphrases are domain-specific words. Since we trained our model on general data, they were not often included in our model's training data. As the model does not have enough knowledge on how to deal with these words, it is important to verify if the respective sentences were translated correctly. In addition to automatically determined keyphrases, users can manually specify further keyphrases for sentence filtering.

Document View A list of all the source sentences in a document and a list of their translations are shown in the *document view* (Figure 4.1 A) (R2). Each entry in this list can be marked as correct or flagged (Figure 4.4) for later correction. A small histogram shows an overview of the previously mentioned metrics. If a sentence is modified, either through user correction or retranslation by the fine-tuned model, changes in the sentences are highlighted (Figure 4.5). Both the metrics view and the keyphrase view are connected via brushing and linking [332] to allow filtering for sentences that are likely to be mistranslated

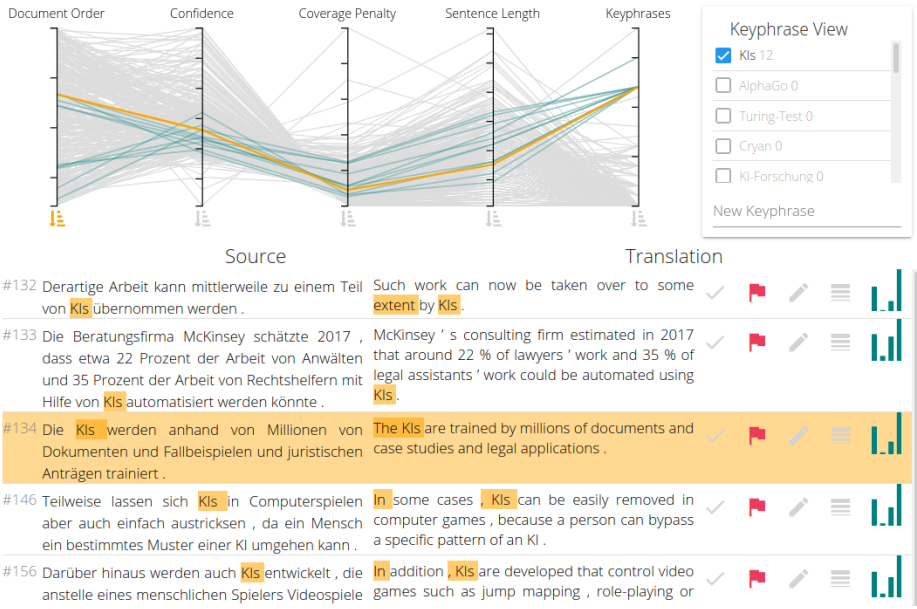


Figure 4.4 — Main view of the system: The document view shows some sentences flagged for correction. Additionally, the keyphrase filter (top right) is active: The metrics and document views show all sentences containing the keyphrase "KIs." Here, "KIs" is never correctly translated to "AIs." This image was created from a document translated by a Transformer model.

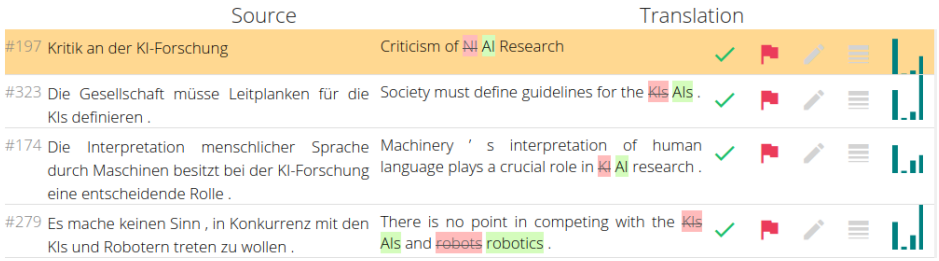


Figure 4.5 — Document view showing user corrected translations by highlighting changes to the initial machine-generated translations. This image was created for a document translated with a Transformer model.

and should be examined and possibly corrected. Additionally, sentences can be sorted into a list by similarity to a user-selected reference sentence. In this list, sentences can be selected for further exploration and correction in more detailed sentence-based views.

Exploration and Correction of Sentences

After filtering and selection, a sentence can be further analyzed with the sentence, attention, and beam search views (Figure 4.2) and subsequently corrected (R4). These views are shown simultaneously to allow interactive exploration and modification of translations.

Note, on the sentence level, we use subword units to handle the problem of rare words, which often occur in domain-specific documents, and to avoid unknown words. We use the byte pair encoding (BPE) method proposed by Sennrich et al. [284] for compressing text by recursively joining frequent pairs of characters into new subwords. This means that instead of using whole words to build the source and target vocabulary, words are split into subword units consisting of possibly multiple characters. This method reduces model size, complexity, and training time. The model can also handle unknown words by splitting them into subword units. As these subword units are known beforehand, they do not require the introduction of an "unknown" token for translation. Thus, we can adapt the NMT model to any new domain, including those with vocabulary not seen at training time.

Sentence View Similar to common translation systems, the *sentence view* (Figure 4.2 A) shows the source sentence and the current translation. It is possible to manually modify the translation, which in turn updates the content in the other sentence-based views. After adding a new word in the text area, the translation with the highest score is used for the remainder of the sentence. This supports a quick text-based modification of a translation without explicit use of visualizations. Currently, changing the translation updates the whole sentence after the modified word. Therefore, we do not support deleting or changing words while maintaining the remainder of a sentence.

Attention View The *attention view* depends on the underlying NMT model. It is intended to visualize the relationship between words of the source sentence and the current translation as a weighted graph (Figure 4.2 B). Such a technique was also used by Strobelt et al. [301]. Both source and translated words are represented by nodes. Links between such words show the attention weights encoded by the thickness of the connecting lines (we use a predefined threshold that users can adapt to hide lines for very low attention). These weights correlate with the importance of source words for the translated words. Hovering over a source word highlights connecting lines to translated words starting at this word. In addition, the translated words are highlighted by transparency according to the attention weights (Figure 4.6 top). While this indicates how a source word contributes to the translation, it is also possible to show for translated

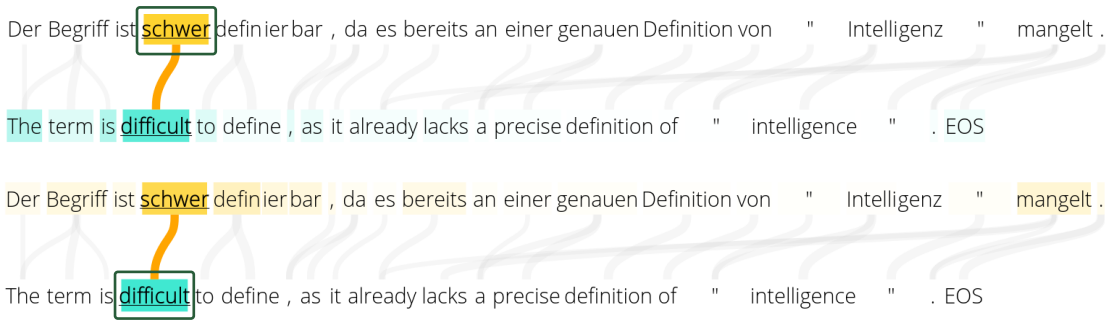


Figure 4.6 — Graph-based attention visualization: (top) when hovering a source word (here: "schwer"), translated words influenced by the source are highlighted and (bottom) when hovering a translated word (here: "difficult"), source words that influence the translation are highlighted according to attention weights. These images were created from a sentence translated by an LSTM model.

words how source words contribute to the translation (Figure 4.6 bottom). This interactive visualization supports users in understanding how translations are generated from the source sentence words. On the one hand, such a visualization helps gain insight into the NMT model, and on the other hand, it helps detect issues in generated translations. The links between source sentences and their translation can be explored to identify anomalies such as under- or over-translation. Missing attention weights can indicate under-translation and links to multiple translated words for over-translation. Section 4.1.4 contains some examples of such problems.

As an alternative, we introduce an interactive matrix-based visualization (Figure 4.2 C). By default, this visualization is not shown due to the limited space on a typical screen, but it can be optionally activated. Visualizing attention weights using a matrix is a common approach initially introduced by Bahdanau et al. [46]. Based on this technique, our implementation is additionally interactive and specifically designed to handle subword units. At the top, the source sentence is presented, and the translation is on the left. The cells show the attention values between the source and translation using a color gradient for the strength of the attention (this correlates to the thickness of lines between words in our graph-based presentation of the attention). We also added special handling for subwords to differentiate them from whole words: horizontal and vertical lines separate all words in the sentences. If there is no line between cells, they represent subwords of the same word. When hovering over a source or target word, the respective words in the other language are highlighted. When hovering over a cell, the words with the largest attention values between the

respective translated word and source word are highlighted for both the source sentence and the translation. This view is linked to the graph-based attention visualization.

We decided to include both visualizations in our system as each has advantages and disadvantages. While the graph-based visualization is vertically compact, long sentences can be challenging to represent on a screen, and often, users have to scroll horizontally. Additionally, there might be some overplotting lines between the source and the translation in the graph-based visualization since there is much less space between the source words and the translation. This makes it challenging to identify where connections start and end, making it difficult to evaluate the attention properly. Despite this, the interactivity and the highlighting of connections enable important information to be explored well. However, providing a general overview of the attention values may be insufficient, especially when dealing with long sentences or when the word order in both languages is very different. For small sentences, this representation works very well. In contrast, the matrix visualization needs less horizontal space (since source words can be drawn vertically). Yet, it requires much more vertical space (usually, it has about the same height and width). Therefore, the general size of the visualization is much larger, and it can be hard to fit on screen for large sentences. However, the attention values can be more easily perceived from the matrix visualization, especially when exploring long sentences. Each attention value is allocated to one cell, and there is no overplotting. Users can inspect the attention in this alternative visualization if the graph-based visualization suffers from too much overplotting by providing the option to switch to the matrix visualization.

As previously mentioned, for the graph-based visualizations, we do not draw all attention connections between source words and their translations when their values are too small. If we added connections for very small attention values, it would be challenging to detect larger attentions due to overplotting (see Figure 4.7 top). Setting a threshold to ignore smaller values helps users identify strong relationships between source and translation words (compare Figure 4.2 (B) and Figure 4.6). When hovering over a word in a visualization that shows all connections (Figure 4.7 bottom), individual attention values of the respective word are still well recognizable. In the user interface, it is possible to change this threshold interactively. This threshold is unnecessary for the matrix visualization since no overplotting problem exists. It would also be possible to add it here to see larger attention values more quickly. However, since each attention value is shown in an individual cell, we decided against this. This allows users to see the whole attention at a glance.

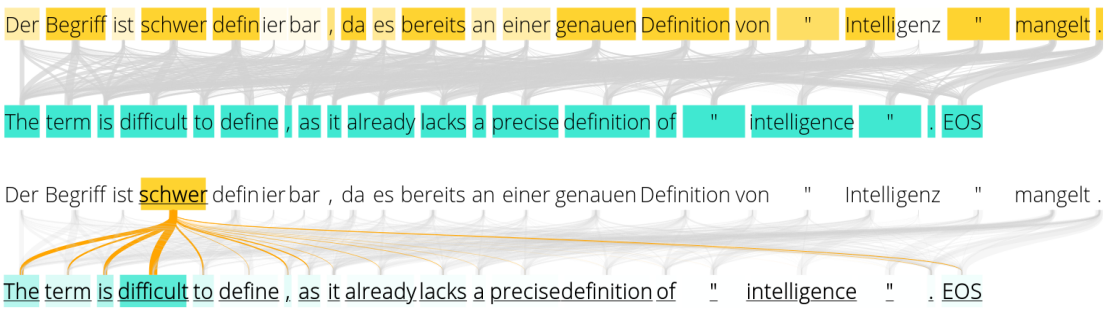


Figure 4.7 — Graph-based attention visualization with adapted threshold setting for connecting lines. Top: Connections show the attention values between all source and translated words. Even for very small attention values, a connection is drawn. This visualization suffers from overplotting. Bottom: Only the corresponding connections are highlighted when hovering over a source word, showing the corresponding attention distribution to all translated words. These images were created from a sentence translated by an LSTM model.

While these techniques specifically employ information of the attention-based LSTM model, we use it in an adapted form for the Transformer architecture (see page 122 in Section 4.1.4). A visualization more tailored to Transformers, also including self-attention could provide additional information. Further models may need different visualizations to generalize our approach, employing model-specific information.

Since the Transformer model has multiple attention layers, it is not apparent which layer should be presented to the user. Our experiments showed that the second to last layer usually showed the best correlation between the source sentences and their translations. Therefore, we select this as the default layer. However, suppose a user is more interested in the behavior of the Transformer model, it is also possible to switch between the different layers to compare the attention distributions and the resulting translations. Figure 4.8 shows the attention values for different layers.

Beam Search View While the attention views can identify positions with mistranslations, the *beam search view* supports users in interactively modifying and correcting translations. The beam search view visualizes multiple translations created by the beam search decoding as a hierarchical structure (see Figure 4.2 D). This interactive visualization can be used to post-edit the translations.

The simplest way of predicting a target sequence is greedy decoding. In each time step, the token with the highest output probability is chosen as the

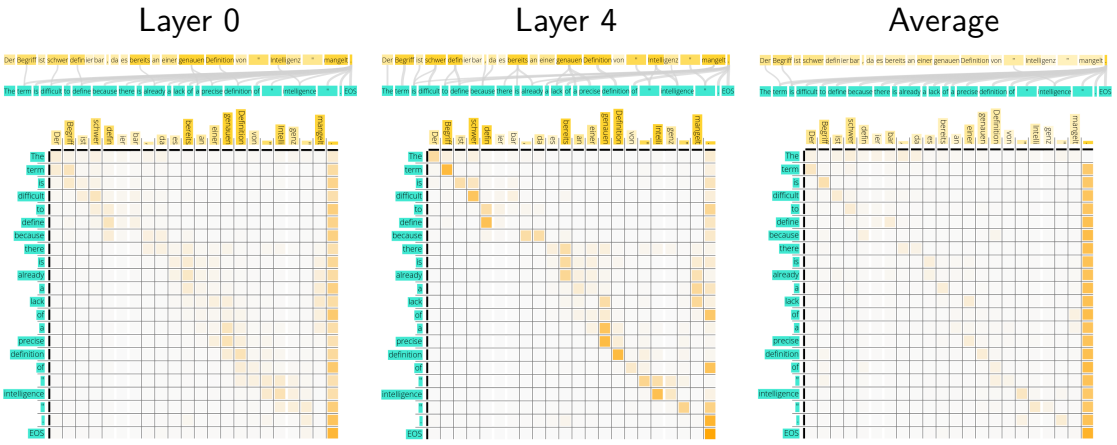


Figure 4.8 — Visualization of the attention of different layers (layers zero, four, and the average of all layers) from the Transformer model. Users can switch between these visualizations. Here, especially the last source word has rather large attention values to almost all translated words. We noticed in our experiments that layer four shows the best alignment between source words and their actual translations. These images were created from a sentence translated by the Transformer model.

next predicted token and fed to the decoder in the following step. This is an efficient and simple way to generate an output sequence. However, another translation may be better overall despite having lower probabilities for the first words. Beam search decoding [135] is a tradeoff between exhaustive search and greedy decoding, often used for generating the final translation. A predefined number (k) of hypotheses is considered at each time step. For each hypothesis, the NMT model outputs a probability distribution over the target vocabulary for the next token. The probability of the final token sorts these hypotheses, and up to k hypotheses remain in the beam. Hypotheses that end with the end-of-sequence (EOS) token are selected to build the result set. Once k hypotheses stay in the result set, the algorithm stops, and the final hypotheses are ranked according to a score function that depends on attention weights and sentence length.

For visualization, we use a similar approach as Strobelt et al. [301] and Lee et al. [202]: a tree structure reflects the inherently hierarchical nature of the beam search decoding. This way, translation hypotheses starting with the same prefixes are merged into one branch of this hierarchical structure. The root node of each translation is associated with a start-of-sequence (SOS) token, and all leaf nodes with an EOS token. Compared to the visualization of a list of different suggested translations, showing a tree is more compact. Additionally,

it is easier to recognize where commonalities of different translation variants lie.

Each translation term is visualized by a circle representing the node and a corresponding label. The color of a circle is mapped to the word's output probability. This helps users identify areas with a lower probability that might require further exploration. It can be seen as uncertainty for the prediction of words. In our visualization, we differentiate between nodes representing subwords and whole words. Continuous lines connect subwords, and nodes are placed closer together to form a unit. In contrast, the connections to whole words are represented by dashed lines.

Our beam search visualization can be used to navigate within a translation and edit it. The interaction can be realized with mouse or keyboard input. The latter method is more efficient for fast post-editing. The view supports standard panning-and-zooming techniques that are especially needed to explore long sentences as they do not fit common displays. For navigation within the tree, arrow keys can be used to move through a sentence, or the mouse cursor can select nodes. If the translation of the current node's child node is not satisfying, the node can be expanded to show some suggestions for correction. If the user selects a suggested word, the beam search runs with a lexical prefix constraint, and our tree structure gets updated. If the suggested words are unsuitable, a custom correction can be performed by typing an arbitrary word that fits better. The user has no limitations here since we use subword units to build words. The number of suggested translations is initially set to three and can be increased by adapting the beam size. Increasing this value may create better translations and provide more alternative translations (Figure 4.9). However, the higher the value, the more information the visualization has to show. By hovering and selecting elements in this view, corresponding elements of the attention views and sentence view are highlighted for reference.

Model Fine-tuning and Retranslation

After correcting the translation of multiple sentences, the user corrections can be used to fine-tune the NMT model and automatically improve the translation of the not yet verified sentences (R5). This approach can be applied repeatedly to improve the document's translation quality, especially for domain-specific texts.

Documents often belong to a specific domain, such as legal, medical, or scientific. Each domain uses a specific vocabulary; the same word can even mean different things in different domains. Therefore, the capability of NMT models to handle different types of domains is very important. Domain adaptation means that

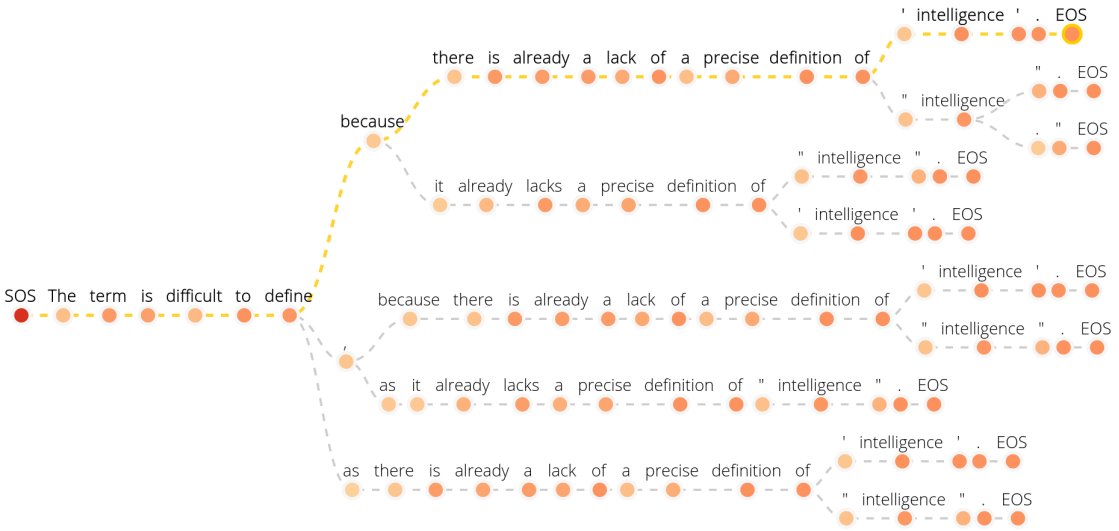


Figure 4.9 — Beam search view with increased beam size. Beam search decoding created ten possible translations, and the beam search representation shows them as a tree to see the similarities between them better. This image was created from a sentence translated by the Transformer model.

NMT models trained on general training data (out-of-domain) can adapt to domain-specific documents (in-domain). This is useful because there is a large amount of general training data, but domain-specific data is rare. Since NMT models require a considerable amount of training data to achieve good translation quality, the out-of-domain data can be used to train a baseline model. The model can be fine-tuned using in-domain data (R5), which usually contains a small number of sentences: we use the user-corrected sentences in our system. This mitigates the problem of training an NMT model where not much data exists for a particular domain. In our approach, we continue training for the in-domain data in a reduced way by freezing certain model weights (for the LSTM-based model, both the decoder and the LSTM layers of the encoder are trained. For the Transformer, only the decoder is trained).

Implementation

NMTVis is implemented as a client-server application. The interactive interface can be shown in a web browser. The client is implemented in TypeScript [58] and runs on the Angular [171] framework. The backend was implemented with Python 3 [321] and the Flask [139] framework. The NMT models are implemented in PyTorch [249], and tokenization of sentences uses Spacy [159]. We apply BPE [284] to the inputs, and the model output is also in BPE vocabulary space. The visualizations are generated with D3.js [65]. The tree layout for the

beam search visualization uses the Reingold-Tilford algorithm [267] implementation in D3.js. Translation quality is evaluated using the sacrebleu [255] library and the Natural Language Toolkit (NLTK) [59].

4.1.4 Evaluation

We evaluate our visual analysis approach from different perspectives using different types of evaluation (see Section 2.7). First, we demonstrate a use case where we automatically translate an article and use the features of our system to enhance the translation quality. Next, we report on the findings of a preliminary user study conducted during the development of our approach. The goal was to evaluate the effectiveness of our concept, using a prototype with an LSTM translation model. In addition, we conducted computer-based experiments to confirm the correlation between our metrics and the translation quality, using both the LSTM and Transformer architectures.

Neural Machine Translation Models

We trained our NMT models on a general dataset: the German-to-English (DE-EN) dataset from the 2016 ACL Conference on Machine Translation (WMT'16) [62] shared news translation task, containing approximately 4.5 million sentence pairs. This is a popular dataset for NMT, used, for instance, by Denkowski and Neubig [102] and Sennrich et al. [285]. In our LSTM-based setup, we used two bidirectional LSTM layers in both encoder and decoder, with hidden size 256 and dropout of 10%. With a batch size of 256, the model (63,219,793 parameters) was fully trained after $\sim$ 16 hours on a single Nvidia GTX 3090.

The configuration used for the Transformer follows the *small* configuration from Vaswani et al. [322] and consists of 6 Transformer layers. A single Transformer layer comprises a feed-forward layer and a self-attention mechanism. Self-attention uses eight heads; linear layers have a size of 2048. A dropout of 10% was used throughout the model. Training the Transformer model (98,167,889 parameters) with variable batch size (maximum of 16384 tokens) took $\sim$ 2 days and 3 hours on a single Nvidia GTX 3090.

We used the bilingual evaluation understudy (BLEU) metric [246] for evaluation. It is a standard evaluation metric in machine translation based on n-gram precision. For each source sentence, a candidate translation from the model is evaluated against a list of reference translations, e.g., translations of multiple human translators.

Table 4.2 — BLEU scores of our models for DE-EN and EN-DE translation on the WMT'16 test sets.

	DE-EN	EN-DE
LSTM	26.95	23.75
Transformer	36.38	33.43

Table 4.2 shows the final performance of our NMT models on the WMT'16 datasets for German-English translation in both directions (English-to-German (EN-DE) and DE-EN) for both the LSTM-based and the Transformer models. Since the goal of our work was not to create the best model, there are others [285] that achieved higher BLEU scores. However, our LSTM model performed similarly to the one by Luong et al. [212]. Our Transformer implementation achieved a better BLEU score for EN-DE translations than both the base and big model by Vaswani et al. [322]. This shows that although there is room for improvement, our NMT models have performance comparable to related systems, making them adequate for our use case.

Use Case

As a typical use case, we take the German Wikipedia article for *artificial intelligence (*Künstliche Intelligenz*)* [340] as a document for translation into English. For translation, we used 358 sentences and headings from the article. In the following, we show how to use our system to improve the translation quality of the document. The following examples were created with an LSTM and Transformer model.

Exploration of Documents After uploading a document to our system (R1), we have a look at the parallel coordinates plot (R2) for our initial translations and the list of keyphrases in order to detect possible mistranslations (R3). The domain-specific term "KIs" occurs relatively often in the keyphrase view. This term is the German abbreviation for "artificial intelligences" and should, therefore, be translated as "AIs." However, none of the translations use the correct term (Figure 4.4). There were also some keyphrases listed that started with "KI" (e.g., "KI-Forschung," en: "AI research") which were wrongly translated. Since the term "KI" (en: "AI") did not appear as a keyphrase, we manually defined it, noticing that it appeared almost 100 times in the document but was never correctly translated. Additionally, one could select and verify sentences with low confidence or a high coverage penalty in the metrics view. Here, we especially notice the under-translation of some long sentences and the over-translation of

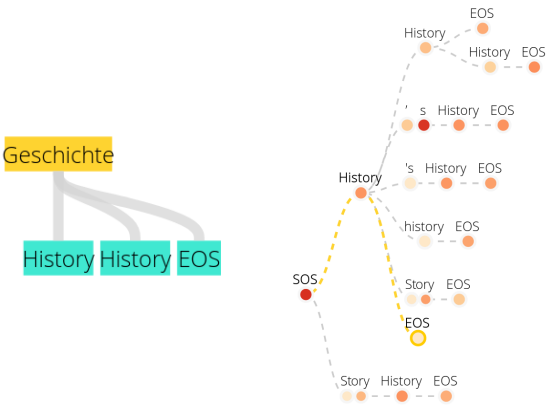


Figure 4.10 — Example of over-translation: "History" is placed twice as a translation for the German word "Geschichte." The beam search view (right) shows possible alternative translations. However, only increasing the beam size to eight shows the translation we would have expected (highlighted in yellow). These images were created from a sentence translated by an LSTM model.

short headings. Additionally, we detected that long sentences often had a low confidence and relatively high coverage penalty (see the filtering in Figure 4.1). After verifying a translation in the document view, users can decide if they are correct (R2). If the users disagree with the translation, they can set a flag (Figure 4.4) to modify the translation later or switch to the sentence-based views to correct it (R4).

Exploration and Correction of Sentences After setting flags for multiple sentences (Figure 4.4) or the decision to explore or modify a sentence, a more detailed view for each sentence can be shown to analyze and improve their translations interactively (Figure 4.2) (R4).

Over-translation is a common issue of NMT [183]. In the attention views, it is possible to see what went wrong by identifying where the attention weights connect the source and destination words. For both models, we notice some cases for very short sentences or headings, and for the LSTM model, some cases with enumerations where commas separate individual words. Figure 4.10 shows for the German heading "Geschichte" (en: "History"), a translation that uses the translated word multiple times. Also, the suggested alternatives use this term more than once. The correct translation is visible only after increasing the beam size to eight, which can then be selected as the correction.

More often, only parts of a sentence are translated, and important words are not considered in our document. Such *under-translation* is shown in Figure 4.11. The German term "Interviewpartner" (en: "interviewee") is skipped in the translation. While this part of the translation is missing, the translated sentence is still correct and fluid. It might be challenging to detect such an error without such attention visualizations.

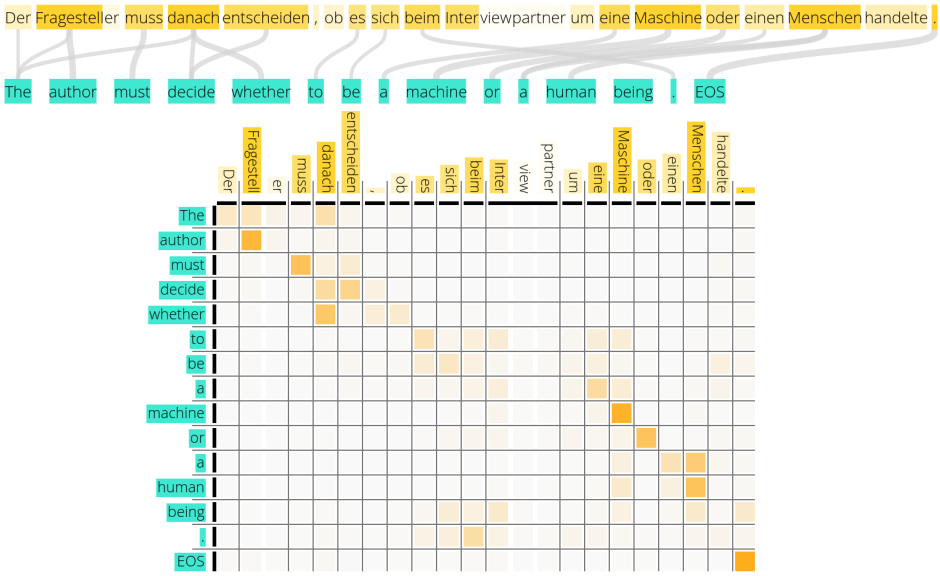


Figure 4.11 — Example of under-translation shown in the attention views: The term "Interviewpartner" (en: "interviewee") in the middle of the sentence is not translated. It can be seen that attention weights are very low for this term. This is visible in both the graph-based and the matrix-based visualization. These images were created from a sentence translated by an LSTM model.

It is even possible that sentences are both under- and over-translated. We found an example where the enumeration of a sentence was not correctly translated. Some enumerated words were translated multiple times, and others not at all. The graph-based attention view showed attention from one word to multiple identical words. Additionally, there were no outgoing connections from every source word. In the matrix-based visualization, the attention values for source words appeared in multiple rows repeatedly and with equal attention strengths.

An example of a wrong translation containing a *keyphrase* is visualized in Figure 4.12. Here, it is also shown that it is possible to select an alternative translation interactively starting from the position where the first error occurs using the beam search visualization. The beam search provides possible alternative translations, but it is possible to manually type what the user believes should be the next term. Here, we manually enter the correct translation. The beam search visualization automatically updates in real-time according to the correction.

Correct translations can also be explored in our approach to investigate the different sentence structures between different languages. Figure 4.13 shows two typical examples of correctly translated sentences. While the first sentence

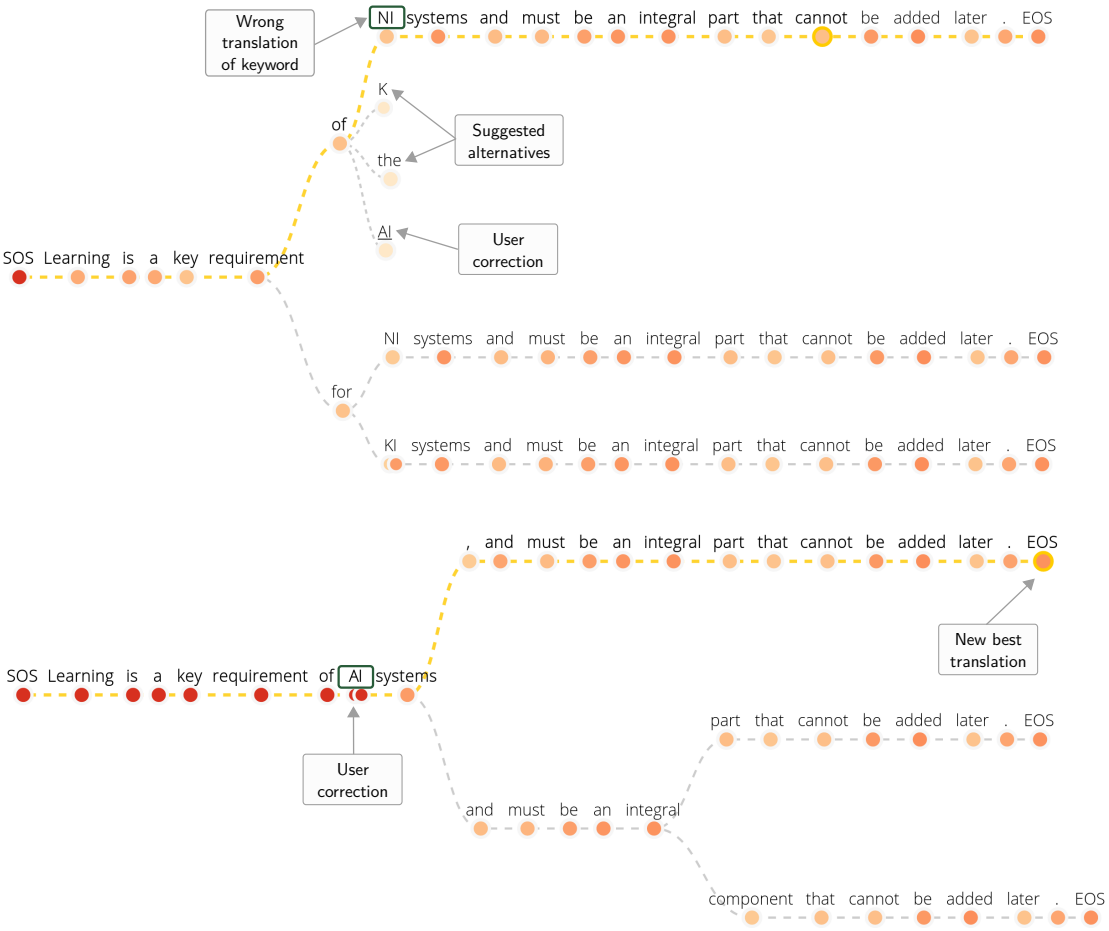


Figure 4.12 — Example of a mistranslated sentence containing the keyphrase "KI" shown as beam search visualization. Top: suggested translation, suggested alternatives, and custom correction. Bottom: updated translation tree for the corrected keyword with new suggestions for continuing the sentence after the custom change. These images were created from a sentence translated by the Transformer model.

has a linear alignment between the source words and their translations, the second example has a different structure in each language. This is visible in our graph-based attention view by the presence of intersecting connections. In the matrix visualization, the same can be seen when most cells along the diagonal are highlighted or when there are some, or even many, outliers. The first example shows that one (sub)word can be translated into multiple (sub)words in the other language. In the graph-based attention visualization, this is visible through multiple out- or in-going connections.

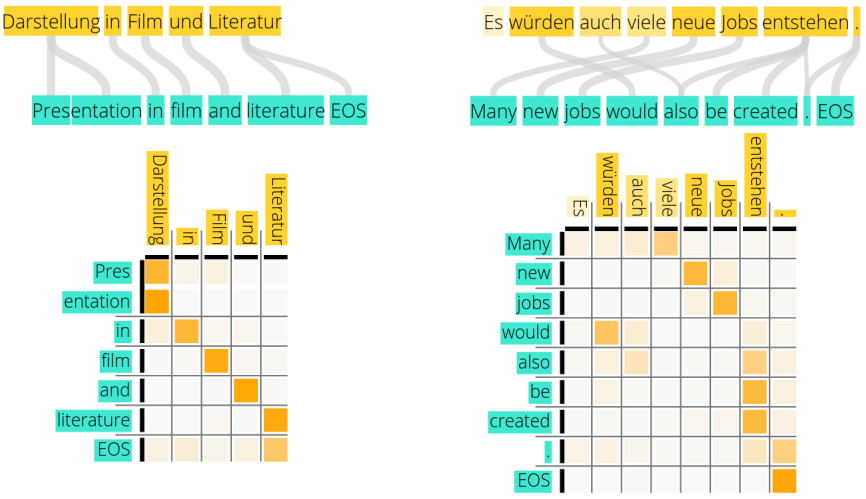


Figure 4.13 — Typical examples of correct translations and the representation of the alignment of words between different languages. In one example, the word order in both languages is the same (left), and in the other, it is different (right). In the graph-based attention view (top), it is possible to see if there is the same word order when there are no or only small line crossings or if the word order is different by larger crossings. In the matrix-based attention view (bottom), the same or a similar word order can be seen by cells with higher values along the diagonal. Outliers along the diagonal indicate a different word order. It is also visible that (sub)words can be translated to multiple (sub)words or vice versa or at least influence multiple (sub)words during the translation. These images were created from sentences translated by an LSTM model.

Finally, it is also possible to change sentences without mistakes. Sometimes, sentences are correctly translated, but different words or sentence structures are used as the current user would prefer for the context of a sentence or to express someone's own style. Again, exploring and selecting alternative words or sentences with the beam search view is possible. If we wanted to use an alternative word like the one suggested by the model, it could be selected or manually entered, and the remaining sentence would be updated accordingly. Further, if we wanted to start the sentence with a different word or restructure the whole sentence structure of the translation, the user could make some adaption at any position of the translation, and the remaining translation would use a possible appropriate continuation.

In all these cases, increasing the beam size to see more alternative translations for a whole sentence (Figure 4.9) or positions within a translation can be useful. This can help choose the best translation or support to rephrase one.

After correcting and accepting multiple translation corrections, the document view shows how translations were changed.

Model Fine-tuning and Retranslation After users correct multiple sentences, they can choose to retrain the current model and retranslate not yet accepted sentences (R5). The model is then fine-tuned using the user's corrected sentences. Afterward, the system translates the uncorrected sentences to improve translation quality. Since our document contains almost 100 times the keyphrases "KI" or "KIs" that were wrongly translated, we retrained our model after correcting only a few (less than 5) of these terms to "AI" or "AIs" (Figure 4.5). After retranslation, the document view shows the difference in the translations compared to before. For the LSTM and the Transformer models, almost all occurrences of "KI" are now correctly translated. The user can look at the changes and accept translations or continue with iteratively improving sentences and fine-tuning the model.

Architecture-specific Observations We initially designed our approach using an LSTM-based model with an attention mechanism. Since other architectures exist to translate documents, we also adapted it and tested its usefulness for the Transformer architecture [322] (R6). This architecture is also attention-based, and we analyzed how well it fits our interactive visualization approach. The general workflow of our system can be used in the same way as the model we initially developed it for: the document and metrics views can be used to identify sentences for further investigation, and sentences can be updated using the sentence and beam search view. The main difference between the Transformer model concerning our approach is the attention mechanism that influences the attention views and some calculated metric values.

The Transformer architecture uses multiple layers with multiple self-attention heads instead of just attentions between encoder and decoder. There are approaches to visualize this more complex attention mechanism [326, 327]. The attention values for Transformers could, for example, show different linguistic characteristics for different attention heads [91]. However, including this in our system would make our approach more complex and not useful for end-users (R7) with little knowledge about this architecture. As a simple workaround to apply our visualization, we discard the self-attention and only use the decoder attention. We explored the influence of decoder attention values from different layers and averaged across all attention heads. Similar to Rikters [271], we noticed that averaging attention from all layers is not meaningful. Using one of the first layers showed similar results. For the final layer, a better alignment could be seen. However, the last token of the source sentence

received too much attention compared to other words (this tendency was observable in every layer). Instead, using the second to last layer showed a similar alignment between source and target words as it is available for the LSTM model. Therefore, we adopt this as a compromise for using our attention views and calculating metric values. By default, our system uses the attention values of the second to last layer, but users can also select different layers and compare their visualizations. An example showing the attention values of different layers from the Transformer model can be seen in Figure 4.8. Here, layer four seems to show the word alignments best, and the results are comparable to the attention from the LSTM model (Figure 4.2 B).

Since there are different approaches and architectures developed for NMT, we could incorporate them as well (R6). Some might provide better support in gaining insights into the model and offer different visualization and interaction capabilities. For others, new ways of visualization will have to be investigated.

User Study

We conducted an early user study during the development of our approach to evaluate our system's concept. We used a prototype with an LSTM translation model. The system had the same views as described before but limited features. A group of 11 voluntary visualization and ML experts from our university (six experts on visualization and five for language processing) were invited to test our system online for general aspects related to visualization, interaction, and usefulness. We aimed to ensure that we considered aspects relevant from both the visualization and the machine translation perspective in our system and to improve our approach. The user study was questionnaire-based to evaluate the effectiveness of the system, understandability of visualizations, and usability of interaction techniques. A 7-point Likert scale was used. In this study, the German Wikipedia article for *autonomous driving* (Autonomes Fahren) [338] was available to all participants. This allowed the participants to explore the phenomena we showed previously. The participants claimed to have good English (mean = 5.1, std. dev. = 0.8) and excellent German (mean = 6.2, std. dev. = 1.7) knowledge. While the visualization experts claimed to have relatively low knowledge about ML (mean = 2.5), the ML experts similarly indicated lower visualization knowledge (mean = 3).

First, participants were introduced to the system with a short overview of the features. Then, they could explore the system freely with no time restriction. Afterward, they were asked to participate in a survey regarding the usefulness

Table 4.3 — Ratings from our user study for each evaluated view are provided on a 7-point Likert scale; mean and standard deviation values.

View	Effectiveness	Visualization	Interaction
Metrics view	5.9 (1.1)	6.8 (0.4)	6.1 (0.7)
Keyphrase view	4.4 (1.6)	6.5 (1.2)	6.3 (1.1)
Beam search view	5.6 (1.5)	6 (1.3)	4.5 (1.8)
Attention view	5.6 (0.8)	6.2 (1.2)	5.9 (0.9)

of our system and its design choices. Additionally, there were free-text sections for further feedback.

The general effectiveness of translating a large document containing more than 100 sentences with our approach was rated high (mean = 5.6, std. dev. = 1.0) compared to a small document containing up to 20 sentences (mean = 4.5, std. dev. = 1.6). The results for effectiveness, ease of understanding and intuitiveness of visualizations, and ease of interaction are given in Table 4.3. The ratings for the visualizations were high for all views. Best rated was the metrics view that additionally had the lowest standard deviation. As not all our user study participants were visualization experts, we noticed that non-experts could also manage to understand and work with parallel coordinates plots. We conclude that our design choice for the visualization of metrics was appropriate. The ratings for interaction were also very high, but there was more variation. Especially the interaction for beam search was rated comparatively low and had the highest standard deviation; two language processing participants ranked it very low (1 and 2) and two (one from each participant group) very high (7). This variation might result from the different learning curves for different participant groups. Since we conducted the user study, we have also improved the interaction in this view. For effectiveness, the keyphrase view had the lowest rating. We believe the reason is that participants could not detect enough mistranslated sentences with this view. However, this might be due to our document provided and may differ from other documents containing more domain-specific vocabulary, as we showed in our demonstration of use cases.

In addition, we asked users for general feedback on our approach. Especially the metrics view received positive feedback. Participants mentioned that brushing and linking are helpful for quickly detecting mistranslations. For the beam search view, one participant noted that the alternatives provided would speed up the correction of translations. For one participant, the attention view helped show the differences in the sentence structure of different languages. Neg-

ative feedback was mainly related to interaction and specific features; some participants suggested new features. Multiple participants noted that exploring and correcting long sentences is challenging in the beam search view as the viewport size is limited. Furthermore, a feature to delete individual words and functionality for freezing areas was suggested. From the remaining feedback, we already included, for example, an undo function for the sentence views. Also, to find sentences that might contain similar errors, one participant recommended showing sentences similar to a selected sentence, and we added a respective metric. Additionally, it was mentioned that confidence scores could be shown in the document list next to each sentence and not only in the metrics view. This would be helpful to quickly examine the confidence value even if the document is sorted by a different metric (e.g., document order); small histograms were added next to each sentence as a quick quality overview.

Quantitative Computer-Based Experiments

Additionally, we performed quantitative computer-based experiments to verify that our metrics correlate with translation quality. We wanted to verify if they support the detection of erroneous and poor translations. Additionally, we were interested in the influence of correcting sentences on the overall document quality. Finally, we examined how our models adapt to domain-specific documents.

For our experiments, we chose the domain-specific Khresmoi EN-DE dataset by Dušek et al. [109] from the WMT'17 shared biomedical translation task [348]. It contains 1500 English sentences with complex medical terminology and German translations. The dataset consists of a test set containing 1000 sentences and a development set of 500.

Metrics to Detect Erroneous Translations In our first experiment, we verified that our chosen metrics (*confidence*, *coverage penalty*, *sentence length*, and *keyphrases*) are suitable for detecting sentences with low translation quality (R3) using both the test and development set consisting of 1500 sentences for evaluation. Scatterplots for the respective metrics against the Translation Edit Rate on Character Level (CharacTER) [331] score (a metric for translation quality on the sentence level where lower values are better) calculated from the translations by the model are shown in Figure 4.14 for the DE-EN LSTM model. Each point represents one sentence. We see that *confidence* has the highest negative correlation ($r = -.35$), whereas *coverage penalty* has a lower correlation ($r = .25$) and *sentence length* a very low correlation ($r = .13$). For *keyphrases*, no correlation is visible. For the Transformer model, the correlations were $r = -.28$, $r = .06$,

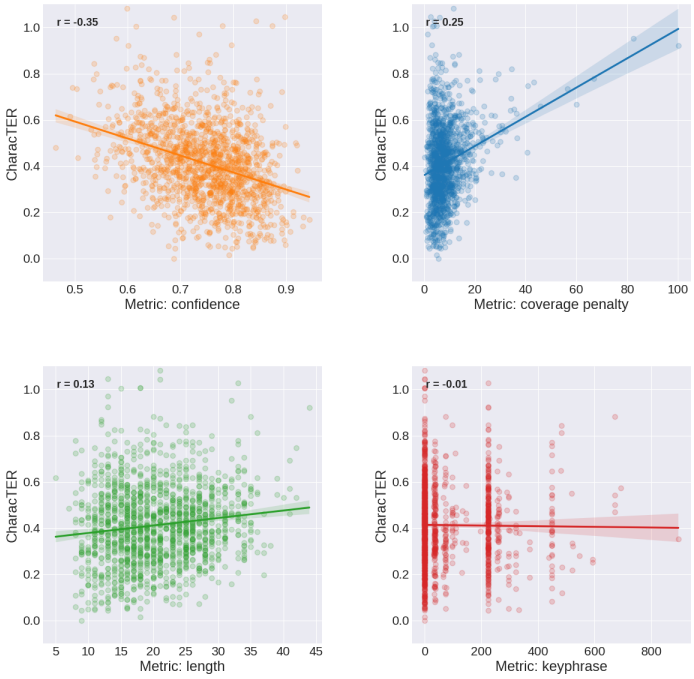


Figure 4.14 — Scatterplots for different metrics about the CharacTER score created for the DE-EN LSTM model. Each sentence is represented by one point. Each plot shows regression lines and the Pearson correlation coefficient r.

$r = .05$, and $r = -.03$, respectively, and therefore much smaller. The correlations for the EN-DE models were similar. Generally, all metrics demonstrate a large variance around the median of metric scores. For these ranges, the metrics are not suitable as predictors. However, high and low scores for *coverage penalty* and *confidence* could be used to detect poor translations. In these areas, the variance is reduced. These outliers, especially the ones for *confidence*, are likely to contain errors. For *length*, there is only a weak correlation with translation quality. Nevertheless, finding the longest or shortest sentence using this metric might still be helpful.

In another experiment, we sorted sentences according to their metric from low-quality to high-quality. Then, we iterated over every sentence and recorded the percentage of low-quality sentences covered so far by removing poorly translated sentences. We considered sentences with CharacTER score larger than one standard deviation above the mean score of the documents to be of low quality. The result for the *confidence* metric showed the best improvement and is highlighted in Figure 4.15 (left) for the DE-EN LSTM model. The result for the Transformer model looks very similar. It can be seen that the coverage percentage increases continuously. The dashed line shows a theoretical baseline of whether the quality of sentences was uniformly distributed. The other metrics are shown for comparison. Here, we notice more variation among the models we used. The DE-EN Transformer model even had a negative impact of *coverage*

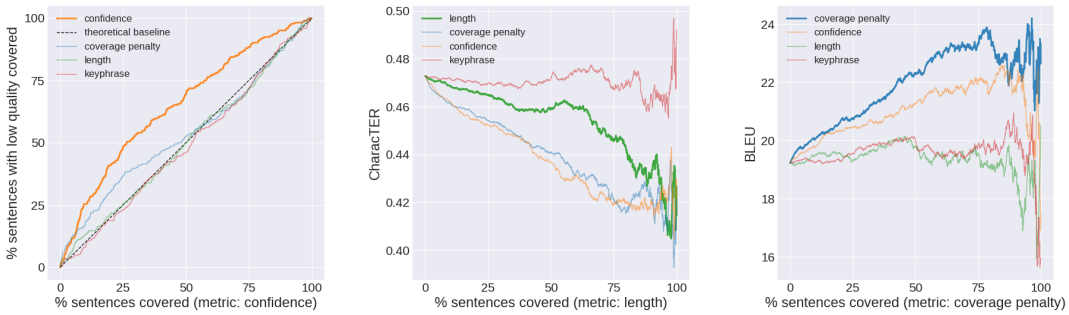


Figure 4.15 — The percentage of sentences with low quality covered after an increased number of sentences on the Khresmoi medical dataset was processed with the DE-EN LSTM model (left). Improvement of average CharacTER scores (middle) and BLEU scores (right) when removing sentences with low quality using the EN-DE LSTM model. In each case, all sentences were sorted according to the different metrics to define the processing order and explore their influence on document quality. In each chart, we highlight one metric.

penalty. However, the *length* metric seemed helpful in this case. Results for the EN-DE models were similar but showed a smaller impact.

Additionally, we observed how average CharacTER and BLEU scores would evolve when removing sentences with lower metric scores. Figure 4.15 (middle) shows for the EN-DE LSTM model the scores *length*, *coverage penalty*, and *confidence* revealing a large improvement of average CharacTER scores for the remaining sentences. Figure 4.15 (right) shows an improvement of BLEU scores, especially for *coverage penalty* and *confidence*. As a note, at the end of this process, when only a few sentences remain to compute the scores, a large amount of variance is visible. Therefore, this part of the visualization is not representative. We obtained similar results for all other models.

Domain Adaption As our system can use corrections to improve the NMT model and to improve translation quality for domain-specific documents, we evaluated for different metrics how the translation quality improved when the models are iteratively fine-tuned (R5). Our initial test in the use case already suggested that significant improvements can be made to domain-specific terms. We tested the improvement of the document on the Khresmoi medical development set and used the sentences of the test set to fine-tune our models and vice versa. We simulated the user corrections by replacing initially generated translations of our model with a reference translation as also done by Peris et al. [252]. Sentences of the first document were sorted according to the metric.

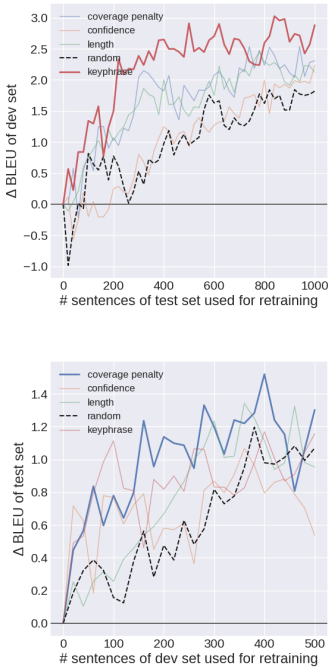


Figure 4.16 — Change of BLEU scores during domain adaption for different metrics using the DE-EN LSTM model on the Khresmoi medical test data (top) and development data (bottom). On the x-axis, the number of sentences that have already been corrected is shown. On the y-axis, the difference between the BLEU score of the translation from the retrained model and the one trained on out-of-domain data is visible. The dashed line represents the result when sentences are randomly ordered. In each chart, we highlight the metric with the best result. On the top, the *keyphrase* score performed best, and on the bottom, *coverage penalty*.

Then, we iteratively fine-tuned our model after replacing the sentences with their ground truths. Twenty sentences at a time were used for fine-tuning. For evaluation, we computed the second document's BLEU [246] score after retranslation. For the LSTM-based models, we observed that using any metric as well as a random ordering of the sentences improved the document quality (Figure 4.16). Especially at the beginning, we noticed a significant improvement for most of our configurations. Interestingly, starting fine-tuning using sentences with a higher *keyphrase* score improved the document quality most in the case of the DE-EN test set (Figure 4.16 top). At the same time, previous experiments showed that the *keyphrase* score had a low impact on selecting incorrect translations. However, not only did the *keyphrase* metric outperform the random baseline (dotted line) over the complete run, but also *coverage penalty* performed well, especially on the development set (Figure 4.16 bottom). *Confidence* usually created an improvement similar to random ordering, while previous experiments showed that this score correlates best when looking for low-quality sentences. Our results suggest that first training our model on sentences containing frequent domain-specific words or with a higher *coverage penalty* score adapts it to a domain better and faster. The results for our Transformer architecture did not improve the document quality. While our tests in the use case showed that the translation of keyphrases could be improved, we could not see an improvement for the BLEU score. We believe this might be due to an already higher BLEU score of the initial translation compared to

our LSTM models. Further research needs to be done to achieve reasonable improvements when adapting a Transformer model, which is beyond the scope of our work.

4.1.5 Conclusion and Future Work

To conclude, we present a visual analytics approach for exploring, understanding, and correcting translations created by NMT. Our approach supports users in translating large domain-specific documents with interactive visualizations in different views, and it allows sentence correction in real-time and model adaption.

Our qualitative user study results showed that our visual analytics system was rated positively regarding effectiveness, interpretability of visualizations, and ease of interaction. The computer-based experiments revealed that using the proposed metrics, particularly the *confidence* metric, can be effective for sentence selection due to correlations of metric scores and translation quality. The *coverage penalty* and *keyphrase* metrics also appear helpful in improving document quality. Additionally, they showed that fine-tuning for a domain-specific document improves translation quality for the whole document.

Currently, users have to use metrics to decide with which sentence they will start correcting the translations. More research has to be done to detect mistranslated sentences better automatically. For example, an additional ML model could be trained with sentences already identified as wrong translations and results could be used for a visualization where users could identify wrong translations. This would be an example of ML4Vis4ML as introduced in Section 2.5.3. A future step should include a more in-depth user study for our target group. For example, we could evaluate the performance of translation tasks by comparing our interactive approach with a manual method.

We believe that our system is helpful for people who have to deal with large documents and could use the features of interactive sentence correction and domain adaption. Comparing the use of our approach for LSTM and the Transformer architecture showed almost no difference. For both, we could successfully interactively improve the translation quality of documents and see model-specific information. We argue that our general translation and visualization process can also be used with further models, while in such cases, some visualization views might need limited adaptation.

4.2 Visual Question Answering

The second approach of this chapter is also located in the field of XAI for NLP: a visual analysis approach for a VQA system (the VQA Explorer) is presented, which is capable of automatically answering questions about an image. In this approach, the input questions are represented as sequences of words and the visualization approach focuses on the scene graphs representing the image content and corresponding internal states of the model during prediction. Since the analysis of sequences was only one part of this project, this thesis only briefly describes the approach. A more detailed description can be found in the work by Schäfer et al. [25] and Künzel et al. [7].

This section is based on the following publications:

- S. Künzel, T. Munz-Körner, P. Tilli, N. Schäfer, S. Vidyapu, N. T. Vu, and D. Weiskopf. Visual explainable artificial intelligence for graph-based visual question answering and scene graph curation. *Visual Computing for Industry, Biomedicine, and Art (VCIBA)*, 8(1):9, 2025, doi: 10.1186/s42492-025-00185-y [7].
- N. Schäfer, S. Künzel, T. Munz-Körner, P. Tilli, S. Vidyapu, N. T. Vu, and D. Weiskopf. Visual analysis of scene-graph-based visual question answering. In *Proceedings of the 16th International Symposium on Visual Information Communication and Interaction (VINCI '23)*, article 25, pages 1–8. Association for Computing Machinery, 2023, doi: 10.1145/3615522.3615547 [25].

(S. Künzel, T. Munz-Körner, and P. Tilli contributed equally to these two publications.)

The images shown in this section were created with the following source code/material:

- N. Schäfer, S. Künzel, P. Tilli, T. Munz-Körner, S. Vidyapu, N. T. Vu, and D. Weiskopf. Extended visual analysis system for scene-graph-based visual question answering. DaRUS, V1, 2025, doi: 10.18419/darus-3909 [28]. (The source code is also available on GitHub: https://github.com/Noeliel/GraphVQA-Explorer.)
- N. Schäfer, P. Tilli, T. Munz-Körner, S. Künzel, S. Vidyapu, N. T. Vu, and D. Weiskopf. Model parameters and evaluation data for our visual analysis system for scene-graph-based visual question answering. DaRUS, V1, 2023, doi: 10.18419/darus-3597 [27].

We propose a visual analysis approach that allows users and developers to gain insight into the reasoning process of a scene graph VQA system (page 41 in Section 2.5.2) using GNNs (page 48 in Section 2.5.2) for the prediction. The VQA system is based on scene graphs [175] used by the prediction model for answering the questions; the scene graphs represent the content of images through nodes and edges. Our visual analysis system (Figure 4.17) facilitates dataset curation and model explanation. It can help users in finding and correcting poor and erroneous scenes. Additionally, internal information about the model during prediction is presented to users to better understand the

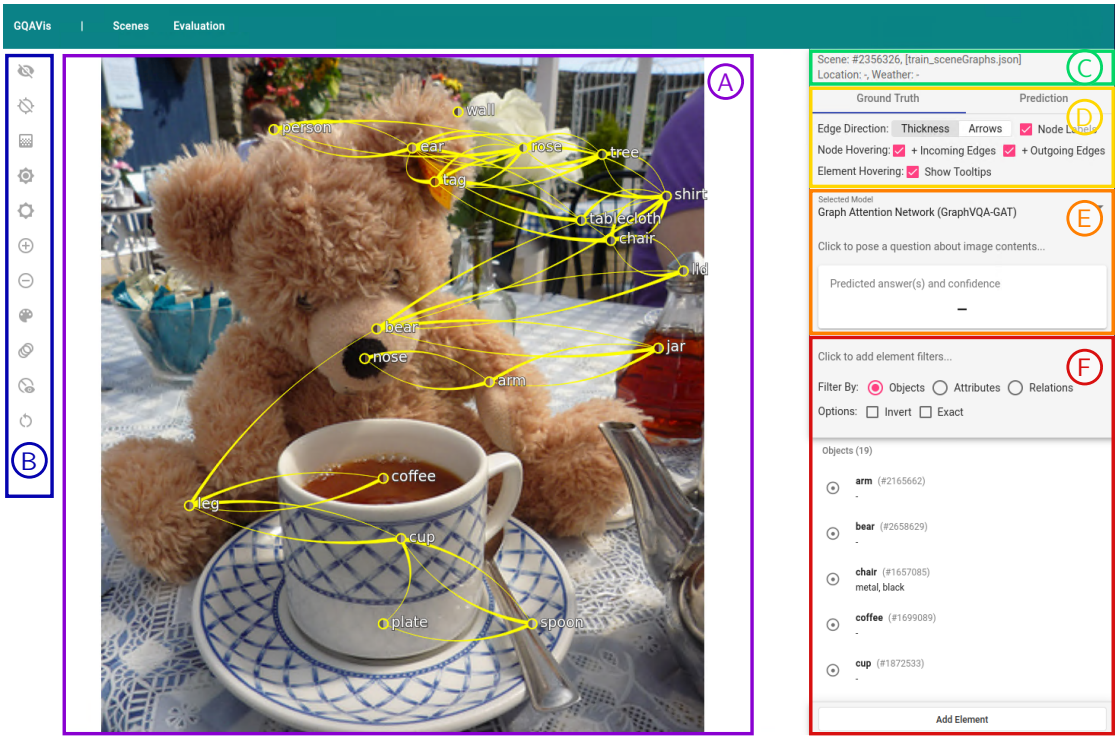


Figure 4.17 — Overview of our visual analysis system: A ground truth scene is shown in the user interface. The scene graph is drawn on top of the image (A) with different options for the visibility of the image and scene graph (B). In the panel on the right side, several information and settings are available: meta-information about the current scene (C), additional settings for the visualization (D), an area for asking a question (E), a list of all objects and relations of the scene graph with filtering options and the possibility to add new elements (F). Background image: GQA dataset [165] (scene 2356326) under CC BY 4.0.

underlying mechanisms and the reasons for correct or incorrect answers. This is similar to the previously presented approach in Section 4.1. Due to the different input data, prediction model, and prediction task, a different visualization method had to be chosen. In the showcased visual analysis system, we integrate the model, scene data, and user input. This allows us to observe information propagation during message passing in GNNs. Users are able to modify both the data of the scene graphs and the input questions to steer the model's attention.

The approach presented here focuses on RQ3: internal states during the prediction process are shown to the user in the scene graph visualization. In this approach, the relation to sequences lies in the input data (questions) and the

data processing through the VQA model. RQ2 is also partially covered since the appearance of visual elements can be customized, which might influence the interpretation of the data.

The major contributions of this approach include the visual analysis method for graph-based VQA systems and an evaluation of the key features of our implemented system. We provide a demonstration of use cases, a quantitative evaluation, and an expert user study. The source code of the system [26, 28] and the evaluation data [27] is publicly available. Preliminary work for this approach was done during the master's thesis of Noel Schäfer [280].

4.2.1 Related Work

In contrast to existing papers dealing with explainability of NLP (e.g., ActiVis [177], LSTMVis [301], RNNVis [228], NMTVis [15], and the system by Garcia et al. [5]; also see Section 2.5.3 and Section 4.1.1), our approach is specifically tailored to graph-based VQA. VQA is a rapidly expanding research area, with many recent methods and datasets that enhance performance [354]. However, like other areas of NLP, the interpretability of the models remains a critical issue. Most existing work focuses on pixel-based visualizations that highlight areas where the model focuses most while answering a question. These approaches are based on different heatmaps plotted on top of an image. They can be applied as attention maps of NNs with attention mechanism [205] or Transformers [164] to highlight areas with the highest attention of specific layers. Goyal et al. [133] use an importance map on the image and highlight important words in the question using, for example, guided backpropagation. Class activation mapping, such as Grad-CAM of CNNs can show the importance of each region of the image related to the decision-making process of the model [328]. Others show error maps to highlight areas that might be erroneous [266] or use multiple maps for different VQA systems to generate a final heatmap [263]. Furthermore, approaches have been designed explicitly to improve interpretability. The model by Norcliffe-Brown et al. [237] learns a graph structure from the question. Subsequently, bounding boxes as nodes and edges between nodes are visualized on top of an image to indicate the most relevant objects and edges in relation to the question, enhancing transparency. Compared to these papers, graph-based VQA does not directly predict based on the image data but on scene graphs [175]. Ghosh et al. [132] generate explanations in natural language from a scene graph and an attention map. However, to our knowledge, no existing work provides a visual analysis approach for VQA based on scene graphs.

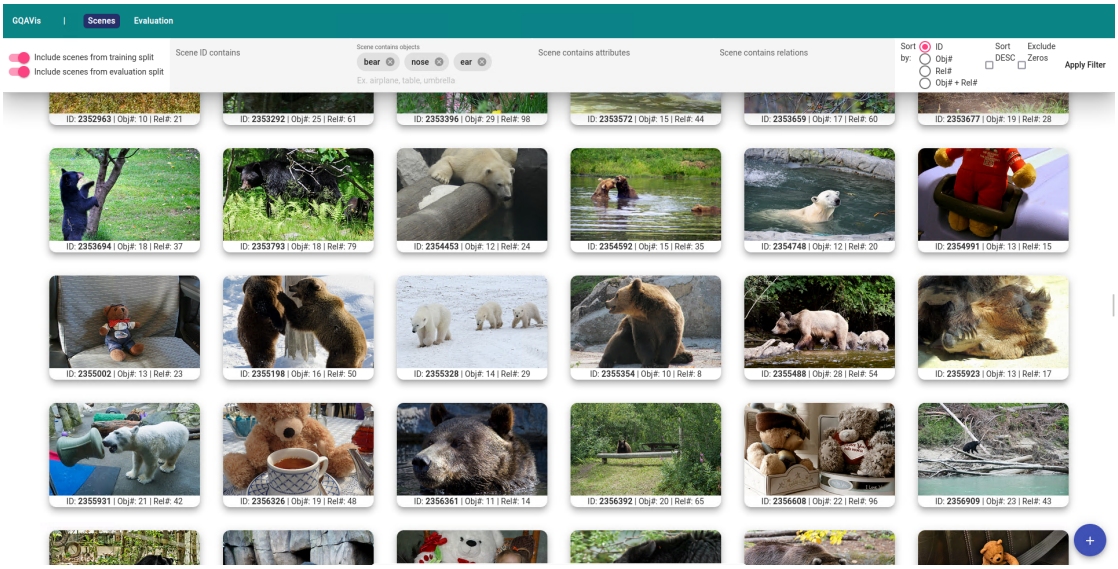
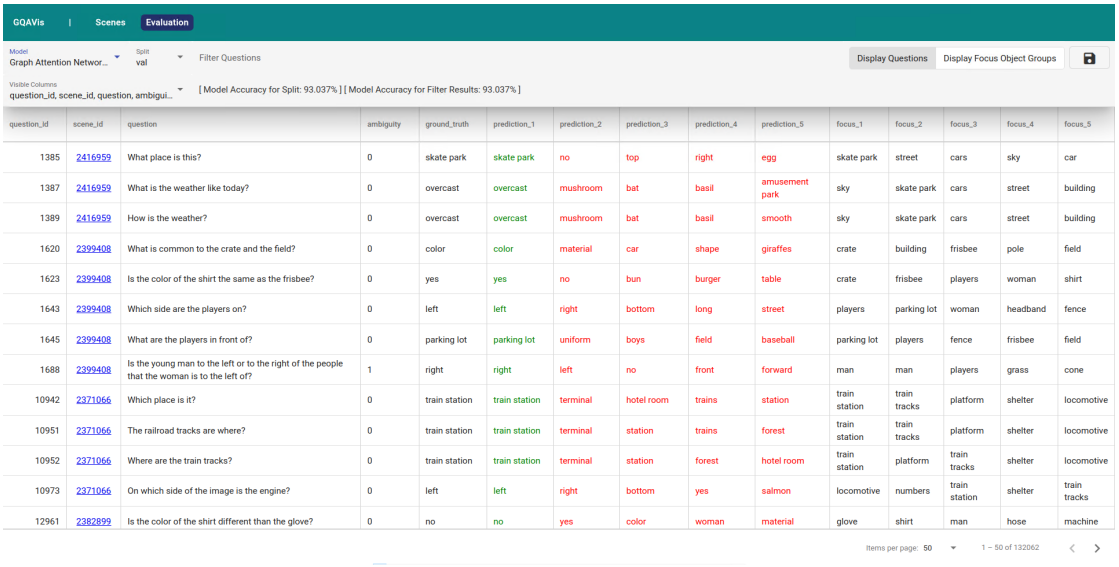


Figure 4.18 — An example instance of the scene browser. Each scene is represented by its image. Image thumbnails: GQA dataset [165] under CC BY 4.0.

4.2.2 Visual Analysis Approach

We build our analysis approach on top of a system designed for graph-based VQA, namely GraphVQA [206]. GraphVQA is a framework that utilizes GNNs and scene graphs to solve the VQA problem. Given an input consisting of a scene graph and a natural language question, the framework produces confidence scores for each possible answer category from a closed set. This approach was developed for the GQA dataset [165], which comprises approximately 22 million questions on 113,000 unique images. The dataset also includes descriptions of images via scene graphs that highlight key objects and their associated attributes (such as color) and contain directed relations among objects in natural language (e.g., relative position). In this approach, the input question is processed with a Seq2Seq Transformer architecture to generate instruction vectors. These vectors are then used during the graph convolutions by the GAT layers to update the node embeddings.

Our visual analysis system consists of three views: The *scene browser* (Figure 4.18) can be used to search and select specific scenes and the *evaluation browser* (Figure 4.19) can help identify erroneous scenes by analyzing prediction results. The view to *visualize individual scenes* (Figure 4.17) with scene graphs allows an analysis of internal states during the predictions and the possibility of improving the scene graphs.



question_id	scene_id	question	ambiguity	ground_truth	prediction_1	prediction_2	prediction_3	prediction_4	prediction_5	focus_1	focus_2	focus_3	focus_4	focus_5
1385	2416959	What place is this?	0	skate park	skate park	no	top	right	egg	skate park	street	cars	sky	car
1387	2416959	What is the weather like today?	0	overcast	overcast	mushroom	bat	basil	amusement park	sky	skate park	cars	street	building
1389	2416959	How is the weather?	0	overcast	overcast	mushroom	bat	basil	smooth	sky	skate park	cars	street	building
1620	2399408	What is common to the crate and the field?	0	color	color	material	car	shape	giraffes	crate	building	frisbee	pole	field
1623	2399408	Is the color of the shirt the same as the frisbee?	0	yes	yes	no	bun	burger	table	crate	frisbee	players	woman	shirt
1643	2399408	Which side are the players on?	0	left	left	right	bottom	long	street	players	parking lot	woman	headband	fence
1645	2399408	What are the players in front of?	0	parking lot	parking lot	uniform	boys	field	baseball	parking lot	players	fence	frisbee	field
1688	2399408	Is the young man to the left or to the right of the people that the woman is to the left of?	1	right	right	left	no	front	forward	man	man	players	grass	cone
10942	2371066	Which place is it?	0	train station	train station	terminal	hotel room	trains	station	train station	train tracks	platform	shelter	locomotive
10951	2371066	The railroad tracks are where?	0	train station	train station	terminal	station	trains	forest	train station	train tracks	platform	shelter	locomotive
10952	2371066	Where are the train tracks?	0	train station	train station	terminal	station	forest	hotel room	train station	platform	train tracks	shelter	locomotive
10973	2371066	On which side of the image is the engine?	0	left	left	right	bottom	yes	salmon	locomotive	numbers	train station	shelter	train tracks
12961	2382899	Is the color of the shirt different than the glove?	0	no	no	yes	color	woman	material	glove	shirt	man	hose	machine

Figure 4.19 — An example instance of the evaluation browser: For each question from the GQA dataset, the top five predictions, along with their prediction scores, are displayed.

Scene Browser In the *scene browser* (see Figure 4.18), all scenes of the dataset can be seen and selected. It is possible to filter scenes based on scene identifiers, object names, attributes, relation names, or the number of objects/relations to be contained in the scene graphs.

Evaluation Browser The *evaluation browser* allows users to analyze the prediction results of all questions in the dataset (training and evaluation data). This view helps users identify question-scene pairs that lead to faulty predictions. It provides two options for exploration: (1) It is possible to explore questions and their prediction results (e.g., prediction results with their confidence scores, ambiguity, or attention scores from the graph node aggregation) in a table. Identifying a false prediction or low confidence score can be used to find corresponding scenes that may need an improved scene graph for better predictions. For example, questions with high ambiguity are likely to be wrongly answered since the model could give a wrong object the highest weight for answering the question. Here, ambiguity refers to an estimate of the potential uncertainty of the question and considers the number of objects in the scene graph the question may refer to by mistake. Users can select an associated scene to further explore model behavior using our inference-connected visualization. (2) The prediction results can be analyzed based on objects and focus object groups in either a table or a scatterplot through performance values. Objects included

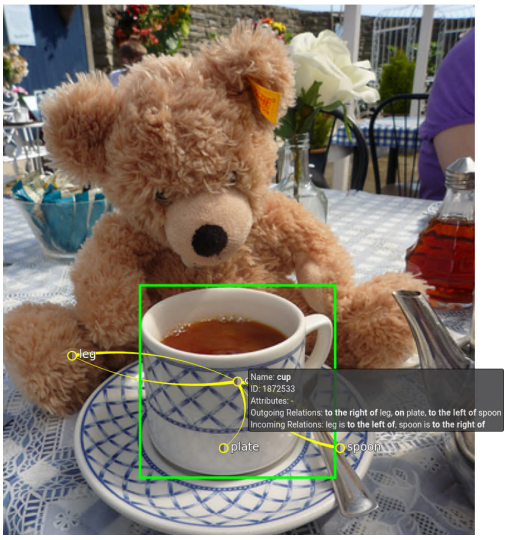


Figure 4.20 — A node is selected from the scene graph. The node is centered within the bounding box of the corresponding object. The bounding box is highlighted in green. Additional information is displayed in the tooltip. Background image: GQA dataset [165] (scene 2356326) under CC BY 4.0.

in a focus object group are taken from the set of graph nodes with the highest attention weights for all questions. In this view, it is possible to find objects that have a low performance.

Scene Graph Visualization In the *scene graph visualization* (Figure 4.17), the scene graph representing the content of an image is visible on top of the corresponding image. The scene graph consists of nodes (for the objects) and edges (for the relations). Graph nodes are colored based on the pixel data of their corresponding scene objects.

As relations among objects are directed, two graph nodes are connected via at least two edges. Relations are drawn as parabolas with vertices offset to the left of the center to avoid overlap for both symmetric and parallel relations between two objects. We show synthesized counter-relations in green and relations natively present in the scene graph in yellow. The visualization provides filter mechanisms for the visualized elements and user interaction to hover graph objects to show an object bounding box and hide additional visual elements. Tooltips contain additional information for elements (Figure 4.20). Additionally, the user interface allows users to modify and improve the scene graphs for data curation and to generate better predictions. For example, it is possible to add and remove objects, relations, and attributes.

The question input field (Figure 4.17 E) can be used to pose an arbitrary question about the current scene to the model. During model execution, various internal model states are recorded to drive visualizations later on. For our GraphVQA integration, we record the global node attention scores (graph gate weights) and the attention weights of edges. Nodes with large attention values end up

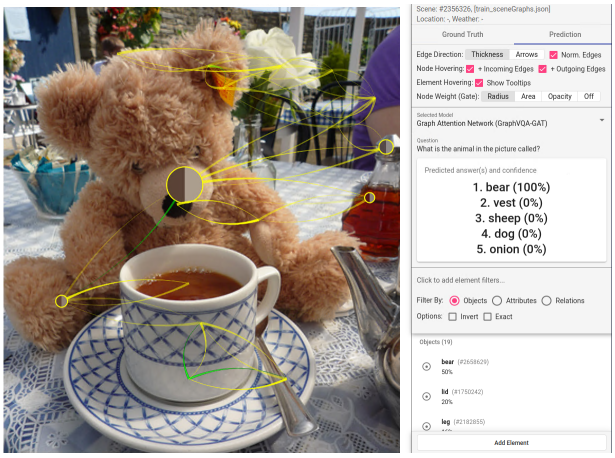


Figure 4.21 — A user asked the question: "What is the animal in the picture called?" The model answers correctly with "bear." In response to the input query, the GraphVQA-GAT model computes weights for each node and edge. The size of the nodes visualizes the graph gate weight for a given node after the final graph convolution. The corresponding opacity value of the edge visualizes the weight of the edges. Background image: GQA dataset [165] (scene 2356326) under CC BY 4.0.

contributing more to the final prediction. They can indicate which information the model chooses to amplify in the context of the question to then base its answer on. In our approach, it is possible to select the GAT [70] model to generate predictions; this is a specific type of GNN (see page 48 in Section 2.5.2). Every edge can be weighted differently in a GAT. Since GNNs aggregate the hidden states of its neighbor nodes, we can leverage this information to highlight certain edges over others. This information enables users to get insight into the information propagation of the underlying GNNs. In our case, GraphVQA stacks five such GAT layers. As a result, users can switch through the attention weights of each layer and observe how strong the contribution of certain edges is for updating adjacent node embeddings. Additionally, model outputs are recorded. The labels belonging to the five highest-ranking categories are displayed together with their confidence scores. In Figure 4.21, the graph gates and edge attention are visualized for an example scene, and the prediction result is shown on the right. Tooltips connected to graph elements contain further contextual information about the elements.

Implementation The presented approach is a web-based interactive system, implemented with TypeScript [58] running on the Angular [171] framework. Python 3 [321] is used for the backend of the system using Flask [139] as web framework. D3.js [65] is used for the visualizations.

4.2.3 Evaluation

A typical use case of our approach is demonstrated in the following; additionally, a summary of our quantitative evaluation and user study is presented. These are different types of evaluation methods as described in Section 2.7. An instance of the GraphVQA-GAT model we trained is used for the evaluation. With an accuracy of around 93%, our model parameters perform similarly but not quite as well as the parameters trained by Liang et al. [206].

Use Case

Poorly labeled scenes often do not result in satisfying predictions by the model. The model can generate false predictions, e.g., when there are incorrect objects, attributes, or relations in the scene graphs. In our tool, finding such scenes, exploring them, and improving their scene graphs for better performance plays a central role. In the following, we show examples of correct and incorrect predictions, how a user can gain insight into the internal mechanisms of the model, and explain the possibility of adding new scenes to the dataset.

Finding Poorly Labeled Scenes Data curation is essential to provide a high-quality dataset. It includes the dataset being constantly maintained and improved. As the first step, we need support finding poor scenes that may require human correction.

In the scene browser (see Section 4.2.2 and Figure 4.18), it is possible to use different filters to search for specific scenes and to sort the data by the number of objects or relations in a scene. Additionally, the quality of individual labels can be verified with the help of the scene browser. When filtering for a specific term (e.g., for an object), the user receives a collection of the scenes that should contain this term. However, we also noticed images where the relevant object was not visible.

The evaluation browser (see Section 4.2.2 and Figure 4.19) has two different views. Using the table for questions, users can sort for questions and corresponding scenes that created wrong (or correct) predictions or prediction results with low confidence. Both can be used to identify the corresponding scenes that may need an improved scene graph for better predictions. Another way to use the evaluation browser is to look for objects that have a low performance. Such objects or object groups can be identified with low prediction quality in a table or scatterplot.

Exploring Scene Graphs The scene graphs can be explored by inspecting the ground truth representation and the result shown after posing a question. In

the latter case, node gate weights and edge attention values of the prediction process are visually presented.

When exploring a ground truth scene with a scene graph (Figure 4.17), it is sometimes immediately visible how well a prediction might perform for some questions. Without deeper exploration, it can be seen if most of the image content is represented as objects and if relations connect these objects. Sometimes, there are not enough objects, and only questions about specific objects can be answered correctly. In other cases, relations between objects are missing.

After posing a question, on the right side of the window (Figure 4.17 E), it is visible what the model predicts, together with a confidence score and the four following best prediction results (Figure 4.21). Additionally, the scene graph shows internal information of the underlying model encoded as node size and edge transparency.

Correct Prediction – Figure 4.21 shows an example of a correct prediction. We asked the question: "What is the animal in the picture called?" The answer was correct ("bear", 100%). Also, the relevant object "bear" has the highest gate weight.

Incorrect Prediction – There can be different sources of errors for incorrect predictions. Besides a poorly trained model, the scene graphs could contain errors or be incomplete.

In our evaluation, we noticed different types of prediction errors that can be created with the model and require different corrections in the scene graphs. The different sources of errors can be roughly categorized as follows:

- **Missing scene graph** – If no scene graph is available for an image, usually no correct prediction about the content of the image can be performed.
- **Missing object** – Missing objects often result in wrong predictions when subject to the question. Adding the relevant object usually generates the expected prediction result. If the question is unrelated to a missing object, it may not generate wrong predictions.
- **Missing attribute of an object** – Node labels for objects alone do not always provide all the information needed about them. Additional information, such as the appearance (e.g., color), is also essential. Missing attributes often result in bad predictions. While the scene graphs look complete and suggest that the correct objects have the highest weights, the answers can still be incorrect.
- **Missing relation** – Another source for poor predictions is missing relations. Only if the connections between objects are available along with

meaningful labels questions about their relationship between different objects (e.g., relative positions or if something is otherwise related (e.g., someone is holding/wearing something)) can be answered. Often, many objects are defined in the scene graph, but connections between objects in the image are missing. In our evaluation, we noticed that different groups of objects or single objects were sometimes not connected. Then, positional questions about these groups cannot be answered correctly.

- **Incorrect label (of an object/relation/attribute)** – Instead of missing objects, relations, or attributes, the already provided information can also be incorrect. Then, the scene graph suggests that something other than the actual content is visible on the image, and answers related to these objects or connections are wrong. This can be wrong object names, attributes, or relation labels. The sources for such errors might be typos, a lack of precision during the annotation, or errors introduced when reusing (e.g., copying) parts of the same or other scene graphs. Similarly, we found examples where labels were not incorrect but also not precise. Such cases also needed correction (e.g., "plant" is a too general term for a "flower").
- **Ghost object** – Sometimes, scene graphs contain objects not available in the image. In other examples, objects were available twice (but with different attributes) that also required correction. Such errors might be introduced due to the same reasons as before.
- **Multiple correct (but unexpected) answers** – A limitation of the prediction model we use is that it can answer every question with only one term. Often, multiple different answers are possible to a question. In such cases, the model selects one of the answers, and the others are usually listed as the following best predictions. In the evaluation browser, such predictions are still shown as wrong since a different answer was expected. Inspecting such a case, a user quickly notices that the model also makes a correct prediction but not the one that was expected from the dataset based on the ground truth data. Additionally, there are cases where one object is very large, and others are very small in the background, and the model referred in the answer to the tiny object instead of the large one in the center. Using the scene graphs, the model cannot differentiate if an object is in the background, very small, or just partially visible in the image. Deleting the nodes of the small items would lose information for other questions and would not be in our interest. Therefore, correcting such incorrect or unexpected predictions is not always possible.

As an example of an incorrect prediction, we asked the question: "What is the bear doing?" (Figure 4.22 left). The model answers incorrectly with "brushing teeth." A correct answer would be "sitting." The answer to another question,

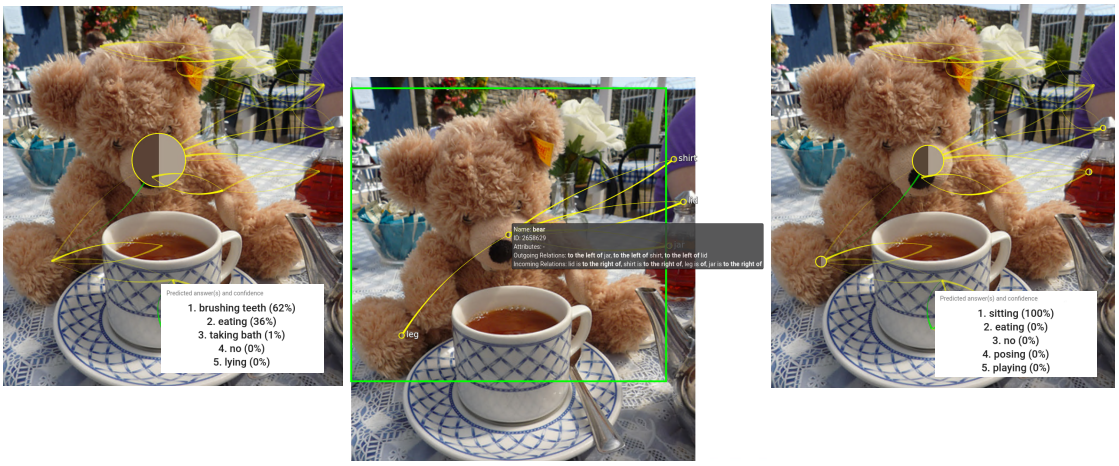


Figure 4.22 — An example where the model answers wrongly to the question "What is the bear doing?" (left). The prediction with the highest probability ("brushing teeth") and the next highest ranked answers are incorrect. After verifying that the object "bear" has no attributes (middle), it is understandable that the model does not know what the bear is doing. After adding the attribute "sitting" to the object "bear," the model answers with a correct response (right). Background image: GQA dataset [165] (scene 2356326) under CC BY 4.0.

where we implicitly suggested what the bear is doing, was also wrong: "Is the bear sitting?" We explored the scene graph with our system. While the node "bear" has the highest node attention (what was expected), the teddy bear has no attributes (Figure 4.22 middle). This is why the model does not know what the bear is doing. We decided to change the scene graph with our system by adding the attribute "sitting" to the node "bear" to improve the prediction. After this modification, both questions are answered correctly (Figure 4.22 right).

Adding New Scenes Our approach also supports adding entirely new scenes to the dataset. Users are free to upload own images, create a scene graph, and pose questions. In a next step, such scenes could be used to retrain and improve the model.

Quantitative Evaluation

To ascertain the relevance of the selected model internals and better understand the implications of the information visualized in our approach, we perform data analysis on model evaluation results to learn more about whether and how intermediate model internals are related to inputs and outputs. We base

our analysis on token co-occurrence as an intuitive measure of node/object significance in the context of a question and a model prediction. We hypothesize that an input question will specify a (group of) target object(s) to which it refers, typically by name or by a distinguishing attribute. This leads to a statistically increasing token co-occurrence count with the input question. Similarly, if the model concludes an answer from a node in the graph, for example, a color or a material, that node will statistically also register more token co-occurrences with the prediction than the nodes of lesser importance that the model did not base its answer on. In summary, we confirmed that nodes with higher token co-occurrence counts with input question or prediction are statistically important in the context of the question or the model's generated answer. The graph gate weights are important intrinsic values of GraphVQA-GAT. Visualizing the graph gate weight per node lets the user see which nodes are focused by GraphVQA-GAT. Our evaluation also revealed that the edge attention score in the final graph convolution does not correlate significantly with the token co-occurrence.

User Study

We performed a user study with experts to assess the usability and usefulness of our visual analysis approach. We invited five experts in visualization and five experts in NLP and DL who volunteered to participate in the study. Participants were from our institution but were not involved in the research for this work except for participating in this user study.

Each expert was assigned two main tasks, performing typical dataset curation: (Task 1) Use the scene browser and evaluation browser to identify erroneous scenes. (Task 2) Correct three given erroneous scenes using the interactive scene graph visualizations. For the first task, the time using the browser and the number of visited scenes were recorded; for the second task the time to complete the task. In questionnaires, all participants provided information about their background, assessed the various features of our system and provided general feedback.

To find two erroneous scenes in the dataset, the evaluation browser is more suited than the scene browser. This conclusion is based on multiple results and observations. First, the experts preferred the evaluation browser. Second, the experts completed the task faster using the evaluation browser. Third, the experts completed the task by visiting fewer scenes, equalling less false positives, using the evaluation browser.

In the second task, the experts corrected faulty scenes. We conclude that our tool is suited for correcting faulty scenes in the VQA dataset. First, the experts

considered our visual representation and visualization to be very helpful for solving this task. Second, the task completion time was short with little variation. Third, all scenes except one were successfully corrected.

In general, the experts considered our tool very useful for scene understanding, scene correction, and VQA transparency, especially for teaching purposes. Apart from minor remarks about usability, the most relevant critique by the experts is the lack of scalability of manual dataset curation. This issue could be addressed in future work, for example, by integrating more automation into the curation process and considering the scalability of visualization approaches [269].

4.2.4 Conclusion and Future Work

We developed a visual analysis system for graph-based VQA models, allowing users to browse through a collection of scenes to identify scene graphs that can be filtered based on user preferences. Once a scene is selected, the ground truth image and a visual representation of the scene graph are displayed. The tool allows users to alter all crucial elements of scene graphs via adding, removing, and editing nodes, edges, and attributes. This process is similar to the one presented in Section 4.1 for NMT: First, users are supported in finding incorrect predictions and can then correct them with the help of interactive visualizations. The system uses the GraphVQA-GAT model for performing VQA tasks and visualizes internal model parameters. Since we deal with a graph classification problem, we enable our tool to visualize scores for each node that determine the importance of nodes. To evaluate our approach, we provided a demonstration of a use case, quantitative measures, and a user study conducted with experts from visualization, DL, and NLP.

There are several ways in which our work could be improved and extended. Since we tested our approach in the context of data curation, there are multiple ways to speed up the process of improving the dataset. An automatic extraction of information from images could be used to spot mistakes and missing objects in the given scene graphs. A verification could compare if labels in scene graphs fit the content on the image and show identified problematic scenes to a user for final correction. Instead, automatically generated scene graphs could be used as a base for improvements by the user. Additionally, after finding faulty scenes, it might be helpful to identify similar scenes that might suffer from similar issues. The process of correcting scene graphs could also be improved. A better recommendation system and copy functionality could speed up the creation of especially large scene graphs. After correcting multiple scene graphs, a fine-tuning mechanism for the model could be implemented to improve its

overall performance by re-training on the updated data samples. Additionally, in other graph convolution models similar information may be available in the final graph convolution. Such models could be investigated and used as a basis in our analysis system. Furthermore, a larger study with additional standardized questionnaires (see Section 2.7) could be performed. This study could also provide stronger statistical evidence for our method.

4.3 Summary and Conclusion

This chapter introduced two approaches in the context of XAI for NLP. Both methods use text sequences as input, process sequential information during training and prediction, and enable a visual analysis of internal mechanisms in ML models. The goals of the visual analysis are to better understand, interpret, and improve results created with supervised ML. Such a visual exploration can help in debugging, data curation, and the decision-making process with the goal of creating better predictions and gaining trust in the models by showing internal information about the models to users. However, many issues still exist for fully understanding and explaining the ML results and internal operations. The approaches presented here show only a part of the relevant information to fully understand the predictions. In addition, the information displayed is not always helpful and meaningful for all cases of the examined data. Some examples showed that our methods can actually lead to a better understanding of the prediction and to improve the data or prediction result. However, there were also cases where the information displayed was not understandable or useful (e.g., the attention values of some layers in NMTVis for the Transformer (Section 4.1) and the edge attention values in the VQA Explorer (Section 4.2)). While the visualized information represents snapshots of the internal states of a model, understanding the whole predictions might be very complex since they are interconnected with many different aspects of the models. Often, when results are not helpful to us for interpretation, the provided information does not align with what we expect when, e.g., translating a sentence or answering a question.

Both presented approaches are mainly centered around RQ3, the exploration of internal states of ML. Our evaluation showed that such an analysis can help better understand why predictions were correct or wrong. Especially for incorrect predictions, it was often possible to see where issues in internal states were present. However, since the underlying prediction models are considered black boxes, we could not provide a solution to understand all aspects. A solution to understand all characteristics should, for example, also include the training data. In an example of the VQA dataset, we could often see why not the correct prediction was made but not why the wrong prediction was made. Here, a reason might be the training data. More research is required to solve and understand more aspects of such black box models. While we explored only sequential data here, the same applies to data and prediction models for data with other properties.

CHAPTER 5

Projection-based Visual Analysis of Sequential and Temporal Data

The previous chapters focused on analyzing temporal data from eye tracking and methods to analyze and better understand ML models containing sequential aspects. Instead, this chapter uses different sequential data (including time series and ML output) as input to create visualizations for the analysis using dimensionality reduction methods. In the following approaches, the input data is multidimensional, possibly high-dimensional, and dimensionality reduction (see page 33 in Section 2.5.1) is applied to reduce the number of features for visualization. The first work of this chapter (Section 5.1) visualizes data of internal mechanisms of a text classification model (also see page 42 of Section 2.5.2) in an interactive visualization system. The second approach highlights problems when using dimensionality reduction to generate visualizations for sequential data and suggests two interactive visualization techniques to avoid misinterpretation of such visualizations.

Note that we use the term *projection* for generalization purposes for any dimensionality reduction method applied to multidimensional data. An explanation for why this term is not always used mathematically correct can be found on page 4 in Section 1.

The first approach presented in this chapter would also fit into the previous chapter since sequential data from ML is analyzed and it contributes to the research field of Vis4ML (see Section 2.5.3). However, a second important

aspect of this work is the analysis method: beyond other visualizations, visual representations for hidden states are created with results from dimensionality reduction. In this approach, we explore a text classification model; exploring other sequential ML data would also be possible. Some master's theses projects I co-supervised used other input data in a similar application: hydrological data [144], turbulence data [48], and the training data of a model used for the prediction of heat conductivity [234].

5.1 Analysis of Hidden States

This chapter presents a visual analytics system to help ML experts analyze the hidden states of layers in RNNs. This technique allows a user to interactively inspect how hidden states store and process information while feeding a text input sequence into the network to perform a classification task.

This section is based on the following publication:

R. Garcia, T. Munz, and D. Weiskopf. Visual analytics tool for the interpretation of hidden states in recurrent neural networks. *Visual Computing for Industry, Biomedicine, and Art (VCIBA)*, 4(1):24, 2021, doi: 10.1186/s42492-021-00090-0 [5].

The images shown in this section were created with the following source code/material:

T. Munz, R. Garcia, and D. Weiskopf. Visual analytics system for hidden states in recurrent neural networks. DaRUS, V1, 2021, doi: 10.18419/darus-2052 [14]. (The source code is also available on GitHub: https://github.com/MunzT/hiddenStatesVis.)

We introduce a visual analytics system (Figures 5.1 and 5.2) to improve the interpretability of RNNs for text classification (page 42 in Section 2.5.2). Here, internal states for the sequential processing of input words are explored with the help of multiple visualizations. As an example, we created visualizations for hidden states with the help of dimensionality reduction. Our approach takes multiple text sequences together with their classification result as input. LSTMs (see page 45 in Section 2.5.2) are used to process the text sequences for the prediction.

In this approach, all RQs are touched. Sequences are visualized in the visual analysis system (RQ1) as colored text, through a path in a projection generated with dimensionality reduction, in a matrix visualization, and a histogram. For some of the visualizations, parameter settings (RQ2) play a role in how the data is presented to the user. This includes a threshold used for the labels in the projection, which show words that might have a strong influence on the prediction and represent large changes in the hidden state. Additionally, the

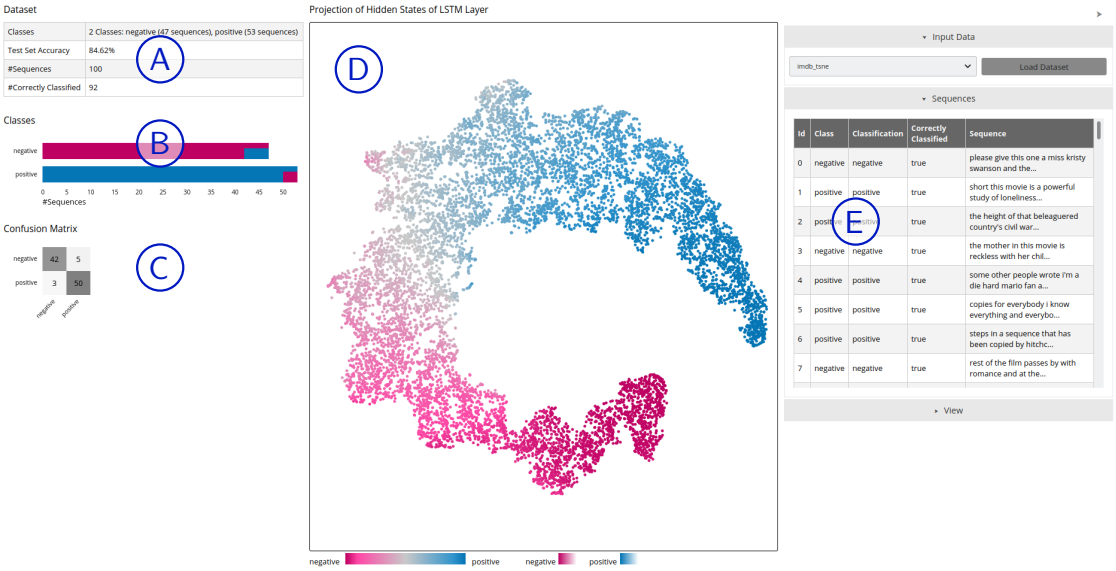


Figure 5.1 — The main view of our visual analytics approach applied to a model trained on the IMDb dataset. Some general information about the dataset is shown on the panels (A) to (E): (A) The number of input sequences, the classes, and how well the sequences were classified. (B) Visualizations of the different classes to compare the number of input sequences that were correctly classified. (C) This information is also visible in the form of a confusion matrix. (D) At the center, the projection of all hidden states produced by the classification model for all input sequences is visible. (E) On the right side, a list of all sequences allows the selection of a sequence for further exploration. Underlying data source: IMDb as available in Keras [88].

size or coloring of visual elements can be adapted. Visual analysis for ML (RQ3) can be used for debugging, primarily to determine why a prediction is wrong or correct (e.g., alternating text sequence in the projection, incompletely processed text sequence, the mentioning of a popular person or some other content not directly related to the sentiment of the text). In this approach, dimensionality reduction (RQ4) is used to analyze the high-dimensional space of hidden states. It is possible to explore the flow of an individual sequence within this space, analyze how the prediction moves in this space, and see the changes in the evolution of the prediction.

This is an extension of the work by Garcia and Weiskopf [129]. We extended the approach by implementing an interactive visual analytics system that facilitates the exploration of the classification process through brushing and linking [332] in multiple coordinated views [273]. The system supports the detection of

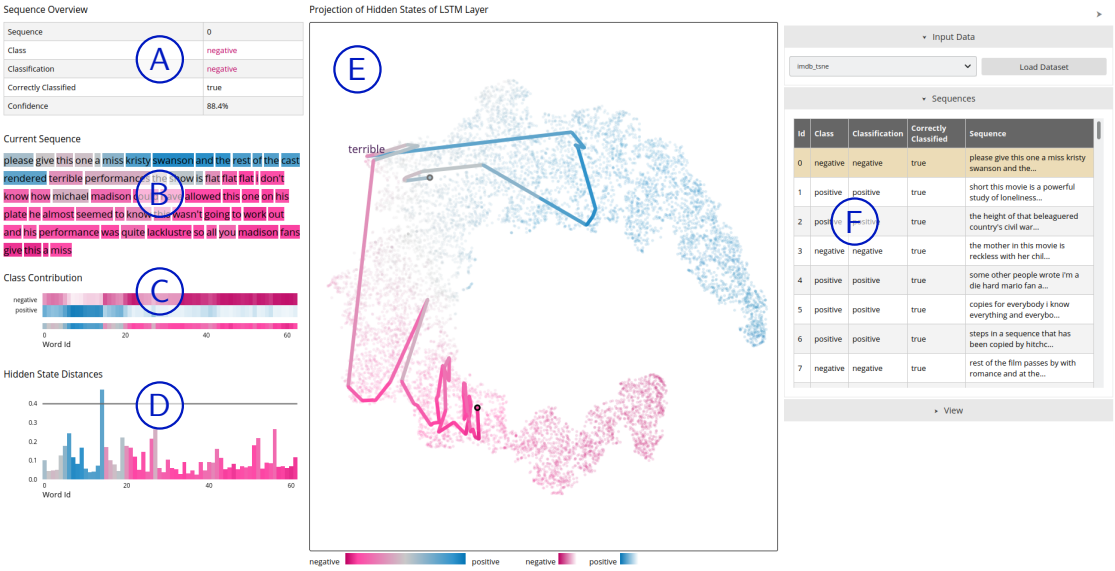


Figure 5.2 — Detailed view for one selected sequence from the IMDb dataset. Different information and interactive visualizations are shown on panels (A) to (D): (A) The sequence ID and information about the classification. (B) The input sequence with each word is colored according to its expected prediction (EP), i.e., what output the model would produce if that word was the last in the sequence. (C) A heatmap matrix displaying how the model's prediction evolves during the sequence processing for each class (top) and overall (bottom). (D) A histogram of the Euclidean distances between the hidden states of the input sequence. (E) The center highlights in the projection the hidden states produced by the sequence, giving insight into how the hidden state evolves over the sequence processing. It is possible to define a threshold for distances to show words in the projection resulting from a larger change in the hidden state; this threshold is also visible as a horizontal line in the histogram. (F) On the right side, the currently selected sequence is highlighted in a list. In the example used for these visualizations, we note how the model first believed the sequence to be classified as positive (blue) during the first few time steps of the input sequence. However, as it obtained more information about the input, it changed its output to a negative value (pink). Underlying data source: IMDb as available in Keras [88].

correctly or incorrectly classified sequences and the process of debugging misclassified sequences to identify why an incorrect prediction was performed and which words contributed to such a prediction. We provide additional information about the Euclidean distances between the hidden states and the words that trigger larger changes. Further, we added visualizations that provide general information about the classification results of the underlying model to

provide an initial quality assessment of the classification of all input sequences. The source code of our system is publicly available [14].

5.1.1 Related Work

Visualization and visual analytics provide interpretability to DNNs at several levels [130]. Therefore, there exist many visualization approaches for DNNs (also see Section 2.5.3). For instance, some techniques have been specifically designed for an analysis of classification models. For example, ClaVis [6] is a visual analytics system used to compare multiple classifiers. Other approaches can be used to explain which input features were taken into account by the model to formulate a prediction [116, 350] and how the decision process of the model transforms the input data into a more abstract representation that facilitates the prediction [130, 157]. In particular, to highlight the differences between the activations produced by elements of different classes, activation vectors of the models can be visualized, as an example, by employing dimensionality reduction [177] and heatmap matrices [265].

Although these methods manage to explain which input features impact the classification, they offer little insight into how the model uses the input features to build predictions. To address this problem, other techniques have focused on an analysis of the internal activation vectors produced by the hidden layers of the network [130, 157]. NNs process inputs by sequentially applying the operation defined by each layer on the input data, generating an activation vector that serves as the input for the next layer. Activation vectors can be seen as a more abstract representation of the input data. They iteratively transform the input data in a way that makes it easier to conduct the classification.

For a deep model to perform well, its hidden layers should produce data representations whose classes are more distinguishable. Ideally, elements from the same class should generate closely related activation vectors in the final layers, whereas elements from different classes should generate different activations. When this is not the case, it is an indication that the model is underperforming and may not be suitable for that application.

Multiple studies have used visualization techniques to analyze activation vectors [177, 265]. They employ techniques such as dimensionality reduction [177] and heatmap matrices [265] to highlight the differences between the activation vectors produced by elements of different classes. By doing so on multiple layers, they can show the decision process of the model and how the input representation evolves layer-by-layer. It also allows the user to identify classes that may be harder for the model to distinguish.

An activation vector analysis is as important for an RNN as it is for other architectures, and the techniques above can also improve the interpretability of these models. However, RNNs have extra complexity owing to the presence of hidden states, i.e., internal memories that store information from previous time steps in the input sequence. When a recurrent model reads an input sequence, its internal hidden states are updated at each time step, generating an activation vector after the entire sequence is processed. Such an analysis is even more challenging when working with models trained for NLP tasks [345]. In a text classification task, some of the input words may have a much larger impact on the hidden states and the final prediction than other words. In addition, some words may produce information that must be stored in the hidden states for a longer period of time than other words. The natures of RNNs and NLP provide new challenges that are not addressed through techniques developed only with an activation vector analysis in mind.

Some techniques have previously focused on improving the interpretability of RNNs trained for NLP tasks. For instance, LSTMVis [300] allows the user to identify patterns in the hidden states when processing a sequence, such as multiple-input words that produce a similar hidden state. Likewise, RNNVis [228] employs a hidden state cluster visualization to correlate groups of similar input words with hidden state configurations.

Although these tools can display patterns in the hidden state and relate them to particular inputs, they do not address open interpretability issues. In particular, to the best of our knowledge, there are no techniques that have addressed the problem of visualizing how hidden state configurations are distributed in a high-dimensional space and how different regions of this space correlate to different prediction values. In addition, they did not evaluate the impact of each input step in the hidden state configuration or, consequently, in the final prediction. Here, we aim to discuss these challenges and introduce a set of techniques that can resolve them.

5.1.2 Interpretability Challenges

The temporal nature of an RNN (also see page 45 in Section 2.5.2) introduces new interpretability challenges when compared to other DNNs. Below, we list three interpretability challenges that, according to our research, must be addressed to achieve better interpretability with an RNN. This is not meant to be exhaustive. Recurrent networks share several characteristics with other architectures, which means that there is a strong intersection between the interpretability issues present in RNNs and those in other models. Herein, we focus on issues specific to RNNs.

Input-to-Hidden-State Correlation When an RNN processes a sequence, it has an initial hidden state vector in each recurrent layer. When the layer receives the next element from the input sequence, this vector is updated according to the information extracted from that piece of information. This behavior allows the model to combine information from previous and future time steps to build a final activation vector that, ideally, should contain the features needed for the prediction task. Hence, the same input step (e.g., the same word in an NLP model) can generate a vastly different hidden state depending on the words that came before it. In addition, the impact of such an input on the hidden state may differ since the initial time steps usually have a higher impact owing to the lack of information of the model at this point. The analysis of the correlation between the input values and their impact on the hidden state is a key element in improving the interpretability.

Hidden State Space Analysis Hidden states are an abstract, high-dimensional representation of the input sequence. However, unlike activations, they are built iteratively during the processing of the input. The layer updates its hidden state at every time step. We can interpret the space of all possible hidden state configurations as a high-dimensional space within which the subsequent layers apply a classification. Analyzing this space and understanding how different configurations are spread over this space are important for increasing the interpretability of the RNNs and answering questions such as whether there is strong class confusion in the configurations and how they evolve during the sequence processing.

Hidden-State-to-Output Correlation A model with a good performance should produce hidden states that are easily distinguishable from the hidden states of the opposing classes, particularly in deeper layers. Because the initial time steps may not hold sufficient information to distinguish between classes, this separability must be built through the input processing. An important interpretability challenge of an RNN is to identify how intermediate hidden states, created during sequence processing, correlate to the prediction and how this prediction, changes when the RNN receives new time steps.

5.1.3 Visual Analytics Approach

Our visual analytics technique addresses the three challenges discussed in the previous section. Our interactive approach consists of multiple views [273], allows brushing and linking [332], and uses a visual analytics approach [182] (see Section 2.1). We show different visualizations for different aspects of the classification, where users can interact with the data provided for analysis. Our

system consists of an overview of all classification data (Figure 5.1) and a more detailed presentation of individual sequences (Figure 5.2).

The overview provides (1) some general information regarding the dataset and its classification results (Figure 5.1 A–C), (2) the projection of the hidden states produced for different time steps and input sequences for the entire dataset (Figure 5.1 D), and (3) a table with the properties of all sequences, allowing the user to select one sequence for further analysis (Figure 5.1 E).

Sequence-based visualizations are composed of multiple coordinated views, i.e., (1) some general information about the classification of the current sequence (Figure 5.2 A), (2) a visualization of a user-specified input sequence colored by the expected output for every partial sequence (Figure 5.2 B), (3) highlighting of the trajectory followed by the hidden states produced through the selected input (Figure 5.2 E), (4) a heatmap matrix displaying how the expected output evolves through the processing of the input sequence selected by the user (Figure 5.2 C), (5) a histogram showing the Euclidean distances between the hidden states of the input sequence (Figure 5.2 D), and (6) a table where the current sequence is highlighted (Figure 5.2 F).

To build these visualizations, we first trained the model (next section) and extracted the hidden states produced in each time step for all test data. Thus, we are left with a dataset of $H = T \times N$ hidden states, where T is the number of time steps, and N is the number of test sequences. Each hidden state $h_i \in H$ is a vector. For each of these configurations h_i, we calculated the expected prediction (EP) p_i. An EP is the output produced by the model if the hidden state h_i were the last time step of the sequence fed into the model. The EP tells us how the model classifies the input until that moment, and consequently, how later time steps modify the model decision. We describe our visualization technique in more detail below.

Classification Overview When loading a new dataset, an overview of the classification quality of the available sequences is presented (Figure 5.1 A–C). Herein, we used two different visualization approaches to show the effectiveness of the classification for different classes. (1) A stacked bar chart (Figure 5.1 B) shows the number of sequences for each class in the dataset and their classification results. Each class has a unique color. The lengths of the bars indicate the number of sequences. On the top, the color of the actual class is visible, and at the bottom, a stacked bar shows the classification of the sequences. (2) A confusion matrix (Figure 5.1 C) quantitatively reports the classification results. Both visualizations show the same information in different ways. We decided to include both of them because the confusion matrix represents the data in a conventional manner, and the stacked bar chart provides a quick overview of

the classification results and links to the other views by using the same color encoding.

Hidden State Projection By projecting the entire collection of hidden states H into a 2D space using t-SNE [319] (or another projection method; see Section 2.5.1 for a selection of approaches), we can analyze how the space of hidden state configurations is shaped (second challenge from the previous section). Herein, we have a set of hidden states $h_i \in H$ that consist of the hidden states of each sequence for each time step, where each h_i is represented as a vector. We then use dimensionality reduction on all hidden states h_i to project them into 2D for visualization. The dimensionality of each hidden state depends on the number of hidden units that can be individually defined for LSTMs. In our experiments, we used t-SNE; however, other dimensionality reduction techniques can also be applied. A discussion on which technique or parameter configuration leads to the best visualization for a particular model is beyond the scope of our work. By coloring the data points according to their corresponding EP, we can analyze how different regions in the hidden state space correlate to the level of confidence that the model has with a given prediction. Figure 5.1 D shows an example of this visualization for a binary classification. Because the network has a single softmax output, our color encoding comprises a range from zero to one. For a binary classification, a prediction of zero (a completely negative review) is denoted by pink, and an output of one (a completely positive review) is denoted by blue. A linear color gradient is used from the first to the second color over gray (where the prediction is uncertain).

Although the data points are colored according to the EP by default, it is optionally possible to differentiate between correctly and incorrectly classified sequences using transparency. In addition, all data points belonging to a sequence can be instead colored by the actual class of a sequence (Figure 5.3). This helps identify regions of hidden states in which the classification is rather certain or may have difficulties.

Input Visualization We visualize each time step of the input sequence, with each word colored according to the EP produced by the model after reading that word (Figure 5.2 B). Hence, the user obtains insight into which time steps are more important for classification and how the EP evolves over the processing of the sequence, addressing the first challenge from the previous section.

Visualizing the Expected Evolution of the Prediction The EP of multi-class classification models is multidimensional, and thus cannot be encoded by a single color. A usual option is to color the data points according to the label with

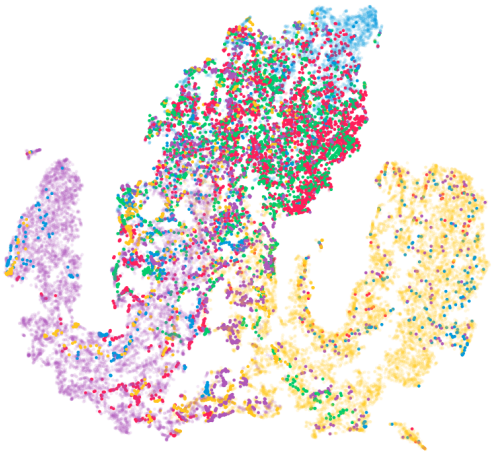


Figure 5.3 — Projection for the Reuters dataset. The data points of the sequences are colored according to their actual class instead of the classification result. In addition, points that belong to correctly classified sequences are more transparent. This visualization helps identify areas where hidden states generate possibly incorrect model outputs. At the top, most sequences were classified as *money-fx* (blue) instead of *crude-oil* (green) or *grain* (red). Underlying data source: Reuters [268].

the highest confidence; however, doing so implies that we will lose valuable information about the strength of such confidence. To avoid this problem, we add a supportive heatmap matrix visualization (Figure 5.2 C) that displays the evolution of the EP vector over the sequence processing (the first challenge from the previous section). Each row displays one possible class, and each column denotes a single time step within the sequence. The color of each square represents the EP of the model for that class at that time step. With our tool, colors with higher saturation translate into a strong belief that the sample belongs to that class, and colors with lower saturation indicate that such belief is smaller. With this matrix, the user can identify whether the model is confident toward the chosen class and at which point in the sequence processing it achieves this confidence. It also allows the user to identify possible confusion among the labels in the dataset. Below the heatmap, we also summarize the class predictions to show the classes with the highest EP at a specific time step.

Visualizing the Hidden State Trajectory The projection of the hidden states (Figure 5.2 E) also allows the user to inspect the evolution of an individual configuration, facilitating the analysis of how each time step impacts the hidden state and the EP. It also provides insight into when the model becomes sufficiently confident of its decision. Our tool adopts the concept of time series projections [45, 318] (also see Table 5.1 on page 171; in Section 5.2 the concept is discussed in more detail), which connects low-dimensional projections of points from a high-dimensional trajectory. Some of these approaches were designed to yield rather smooth curves [45, 155], for example, by including piecewise Bézier curves. However, our variant of the time series projections is purposely

non-smooth (similar to the approach by Molchanov and Linsen [230] and van den Elzen et al. [318]) because we give the user a choice to specify an input and to visualize the specific points reached by the hidden state sequence produced by the input. We color code the trajectory according to the EP of the starting point of each line segment. Because the projection uses the same colors, it may sometimes be difficult to discern the trajectory. Hence, we provide the possibility of using gray levels instead to represent a temporal development. When showing a trajectory for a sequence, the transparency of all data points that do not belong to this sequence is increased. This technique addresses both the first and third challenges described in the previous section. This visualization and all other sequence-based views are linked to explore the relationship in different views when hovering over visual elements. Moreover, a tooltip for each element shows additional quantitative and text-based information about the classification up to the current word or the transition between two words.

Visualizing the Distances between Hidden States In a bar chart, the Euclidean distances between hidden states are shown for the input sequences (Figure 5.2 D). In particular, larger distances (which we also refer to as *jumps*—a term we also use later in Section 5.2 as a reference to large distances in the visualizations of dimensionality reduction results) often push a classification further toward higher confidence in one specific class. They are usually associated with words that are common to the corresponding class. We provide a threshold such that each word associated with a jump is shown along the hidden state trajectory (see next paragraph).

Visualizing Words in the Hidden State Projection It is optionally possible to show words that produce jumps between hidden states as an overlay on the hidden state projection. The corresponding words often correlate with words that are specific to one class. Similarly, the exploration of larger changes in time steps in the EP for specific classes might have a similar impact. Such changes are also visible in the input and heatmap visualization when the saturation of the color or the color itself changes. Using a threshold, the words for these changes can also be shown on top of the projection. In addition, words for changes between classes (e.g., from positive to negative) can be added. Corresponding words may also be specific to a certain class because they change the expected classification up to this word. Often, the words may fall into multiple categories because they all express a change or make a large contribution to the prediction result. Showing these words for all sequences simultaneously for one of the categories also revealed some clusters. For example, words triggering larger

changes for hidden states may have hidden states similar to other words creating a similarly larger change.

Our visualization is sufficiently generic, it can be used in the analysis of different types of recurrent architectures, such as gated recurrent unit (GRU) [86] and LSTM [156] (see page 45 in Section 2.5.2 for more details).

Implementation Data preparation and training were conducted using Python 3 [321] and TensorFlow [29]. Projections for t-SNE were generated using Scikit-learn [250]. Although all images in this section were created using t-SNE, it would also be possible to use other projection methods instead (compare Section 2.5.1 for alternative dimensionality reduction techniques). For our interactive visualization system, we generated a web interface implemented with JavaScript and Python using the Flask [139] framework; the visualizations were generated with D3.js [65].

5.1.4 Use Cases

To evaluate our visual analytics approach, we demonstrate some examples of how it can retrieve insight from models trained with two text classification datasets developed for sentiment analysis tasks (page 42 in Section 2.5.2), the Internet Movie Database (IMDb) dataset [213], and the Reuters dataset [268]. The IMDb dataset [213] comes with a binary classification problem in which every input is a text sequence containing a movie review that is either positive or negative. For this study, we used the IMDb dataset version available in Keras [88]. We trained a model containing a single LSTM layer with 100 units and an output layer with a single sigmoid activation unit, achieving an accuracy of 85%. We opted for a simple model because our goal is to visualize the impact of the recurrent layer in the model classification, and adding further layers would require including them in the analysis.

By contrast, the Reuters dataset [268] comprises thousands of articles on economics that can belong to one of more than 40 classes. For simplicity and easy visualization of the results, we trained our model using only the five most frequent classes (*grain*, *crude-oil*, *money-fx*, *acquisition*, and *earns*). For this task, we trained the model in a similar fashion as the previous model, using a single LSTM layer and an output layer containing five softmax output neurons, one for each class. After training, our model achieved a test-set accuracy of 93%.

Below, we demonstrate how our method can be used to conduct analytical tests in the aforementioned models. For the IMDb dataset, we used the first 100 sequences as input for our visualizations, and for the Reuters dataset, the first 500 sequences.

Binary Classification

Using our approach, we can explore the classification of binary data. In the following, we provide some insights into the IMDb dataset.

Classification Overview The overview visualization shows the relationship between the number of positive and negative reviews in the input data and how well they were classified (Figure 5.1 B and C).

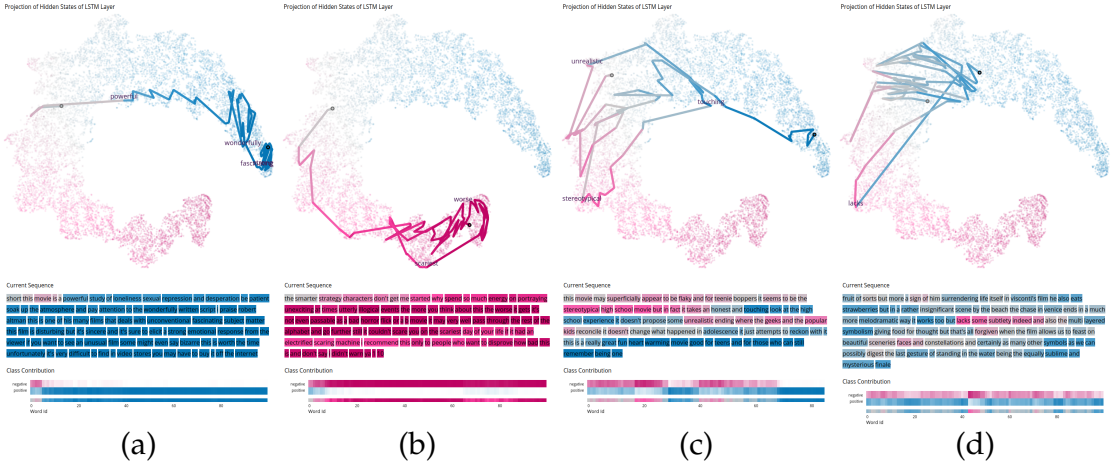
Hidden State Space The 2D hidden state projection allows users to analyze how the RNN models the high-dimensional hidden state space. In Figure 5.1 D, we note that the model creates what seems to be a low-dimensional manifold embedded in the high-dimensional space, which continuously moves from completely negative reviews (pink) to completely positive reviews (blue).

Correct Classification Our tool facilitates an analysis of how the EP evolves through the sequence processing by displaying the trajectory of the configuration within the hidden state space and by displaying, with color encoding, the changes in the EP values. Figure 5.2 and Figure 5.4 show the examples of correctly classified reviews. These include typical reviews that were predicted as positive or negative with high certainty, reviews that jump between the two classes, and reviews in which it is not entirely certain throughout the prediction whether the classification should be entirely positive or negative.

Typical positive and negative classification results are shown in Figure 5.4 for the images on the left. The trajectories in the hidden state projection start in a neutral region and then almost immediately jump toward the very positive or negative regions. This behavior is also visible in the heatmap visualizations.

Figure 5.4 (c) shows an interesting insight uncovered by the tool for another sequence. Although the review is positive, it is written with several negative words appearing in most of the sequence, which makes the model alternate its hidden state between a positive and negative EP. Only by the end of the sequence does the model become certain that the review is indeed positive, and the hidden state converges to a region of a highly positive prediction.

Figure 5.4 (d) shows an example of a review that is not very positive or negative. In between, although there are some rather positive or negative words, the review stays rather neutral overall, and the model struggles to classify toward a single direction. Because reviews can be also neutral with only a slight tendency toward positive or negative, such sequences may also be problematic for humans to clearly classify toward one direction.



(a) (b) (c) (d)

Figure 5.4 — Different examples of correctly classified sequences from the IMDb dataset. From left to right: (a) Typical example of a positively classified movie (blue). The first word that clearly classifies the sequence as positive is "powerful." Afterward, no word suggests that the movie might have been negatively rated. The words "wonderfully," "strong," and "fascinating" result in particularly larger jumps in the hidden state space. (b) A typical example of a negatively classified sequence (pink). Right from the beginning, this review was classified negatively. (c) The input starts with several negative words (pink), making the model alternate between hidden states with high and low EPs. Toward the end of the review, the intent of the writer becomes clear, and the model settles for a positive output (blue). (d) A positively classified movie with uncertainties. At the beginning and end, there are more indications of a positive rating (blue). However, most vocabulary feels more neutral than highly positive. In the middle, there is also a negative statement (pink). Underlying data source: IMDb as available in Keras [88]

By correlating the time steps with the hidden states created by them, our technique can identify undesired biases in the model. Biases occur when the decision process considers non-representative features that an expert does not consider if manually conducting the task. For instance, in Figure 5.2 E, the EP of the model jumps to a highly positive value when the model reads the name of an actress, i.e., Kristy Swanson. This is not the desired behavior because there is nothing in that sentence up to that point that indicates a positive review, and ideally, the model should only consider the sentiment of the reviewer, and not whether a particular actor participated in the movie.

In all of these examples, some words are shown along the trajectories. These words are the result of the larger distances of hidden states when processing a sequence. Such distances for a whole sequence, for example, are visible in

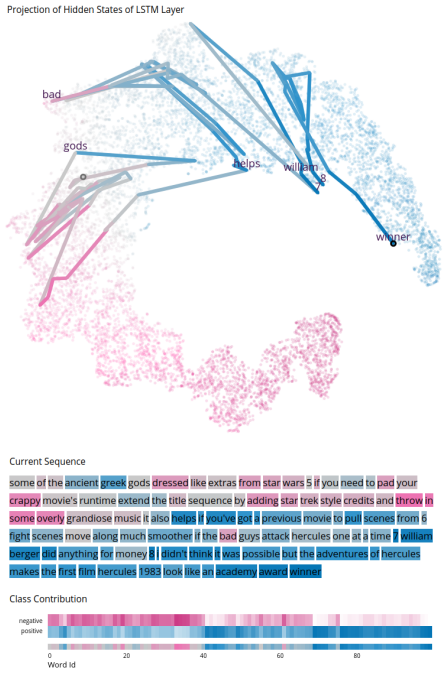


Figure 5.5 — Example of a wrongly classified sequence from the IMDb dataset. For the first part of the sequence, the classification is rather negative (pink). Only in the second half of the sequence does the classification become positive (blue). The last words in particular contribute to the incorrect classification, where the name of an actor appears, and afterward, the words refer to another film that was rated better compared to the current one. Underlying data source: IMDb as available in Keras [88].

Figure 5.2 D. A threshold (shown by the horizontal line) is used to show only words that result from a larger change. This visualization helps explore the strength of the change along the sequence. It also shows that the lengths of the lines in the projection do not always correlate with the distance in the original space owing to the nonlinear projection method applied. For example, this is visible when comparing the long blue line on the top in Figure 5.2 E to the pink line on the left. Although the pink line suggests a large jump, the histogram shows that the distance between the corresponding hidden states of the blue line is larger.

Incorrect Classification An example of an incorrect classification is shown in Figure 5.5. Although this review was negative, it was misclassified as positive. An analysis of the visualizations showed that, at the beginning, the confidence toward a negative classification was higher. However, in the second half, the classification switches to a positive classification. This occurs because the name of an actor appears, and the last part refers to a different movie that was described with more positive words than the actual movie the review refers to. After the last words (“academy award winner”), the model has no chance to change the final prediction.

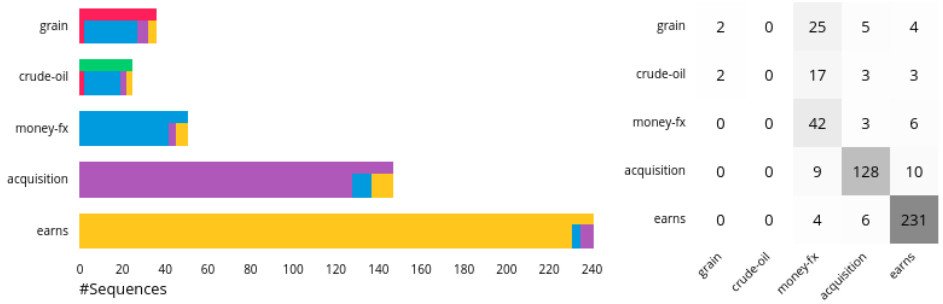


Figure 5.6 — Classification result in the form of stacked bar charts (left) and a confusion matrix (right). The visualizations show that there are many more sequences available for the classes *earns* and *acquisition* compared to the others. In addition, it is clear that sequences of the classes *earns*, *acquisition*, and *money-fx* were well classified, whereas the model had problems with *grain* and *crude-oil*. Underlying data source: Reuters [268].

Multi-Class Classification

In addition to a binary classification, our approach supports a multi-class classification. It can be applied similarly with some adaptation regarding the use of colors. Next, we report some insight for the Reuters dataset [268].

Classification Overview The classification overview (Figure 5.6) shows the different classes with their corresponding color encoding, the number of test sequences available for each class, and how well the sequences were classified. It is clear that the frequencies for the classes vary substantially: *earns* and *acquisition* are frequently included, whereas *grain*, *crude-oil*, and *money-fx* are not. In addition, the correct classification achieved much better results for *earns*, *acquisition*, and *money-fx* than the others, which were mostly classified as *money-fx*. This provides some initial insight into the types of sequences that should be further analyzed.

Hidden State Space and Heatmap Visualization The hidden state projection visualization has certain limitations when handling multi-class classification models. In particular, it cannot properly display how each data point relates to each class. Here, the output of the model is a multidimensional vector in which each dimension represents the likelihood of the input being from a certain class. To mitigate this problem, we developed a supportive visualization in which data points are colored according to the more likely class generated by that hidden state. Hence, the color encoding in the projection simply represents the predicted class, without referring to the strength of the confidence in

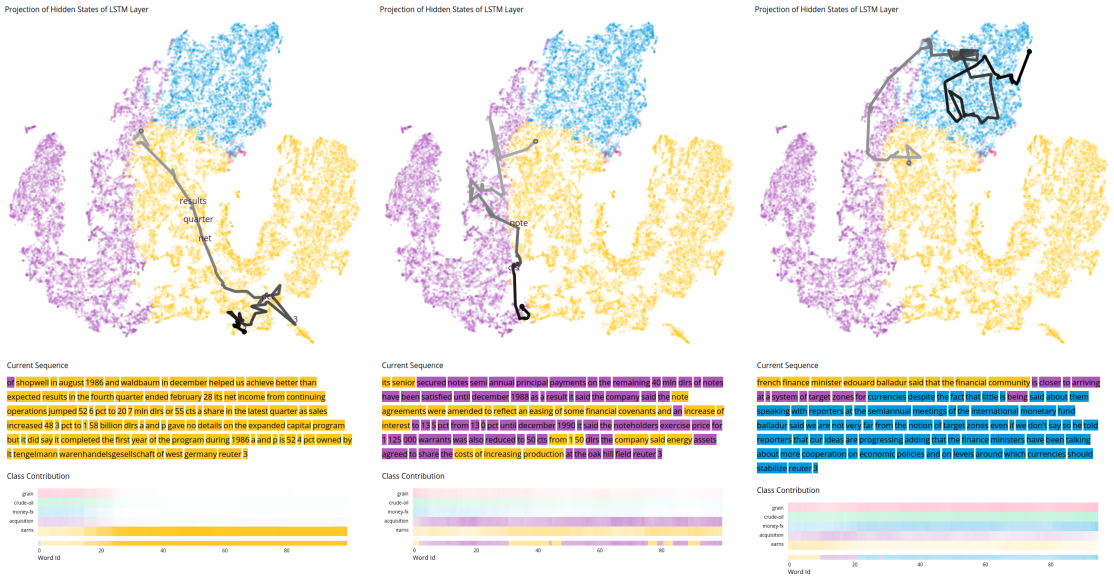


Figure 5.7 — Examples of correctly classified sequences in a model trained for multi-class classification. The projection provides insight into the class distribution within the hidden state space. The heatmap matrix supports the analysis by displaying the evolution of the EP over sequence processing. Hence, the user can better identify at which time steps the model was able to distinguish between classes. In the visualizations on the left, the model is certain for the classification of *earns* (yellow). In the visualizations in the middle, the model is uncertain between *earns* (yellow) and *acquisition* (violet), and both classes have a similar EP while processing the sequence. In the visualizations on the right, the model is uncertain among all five classes, and the EP for *money-fx* (blue) is only slightly larger compared to the other classes. Similar cases are likely to be misclassified. Underlying data source: Reuters [268].

the prediction. Only the color encoding in the heatmap matrix shows the confidence in each individual class, using a color gradient ranging from white (low confidence) to the respective color (high confidence). In the projection visualization, it can be seen that the classes *earns*, *acquisition*, and *money-fx* dominate and build different regions (Figure 5.7). To allow the user to visualize how the EP of a hidden state differs among the possible classes, the heatmap matrix visualization displays the evolution of the EP over the processing of an input sequence.

Correct Classification Figure 5.7 shows multiple examples of correctly classified sequences from a model trained with the Reuters dataset. In Figure 5.7 (left), we notice in the projection visualization that the hidden state sequence starts in

a central location where classes are not clearly distinguishable. However, as the sequence processing progresses, the hidden state moves to a region with more certainty toward one of the classes (in this case, *earns* (yellow)). This conclusion was supported by the heatmap matrix visualization. It should be noted that the model does not distinguish any class until time step 15. At this point, the model begins to converge toward the correct class. After the 20th time step, the model does not significantly change its EP until the end of the sequence, which leads us to believe that this sub-sequence contains sufficient information for the model to make a decision toward that class. Figure 5.7 (middle) shows an example in which the model is uncertain whether the sequence should be classified as *earns* (yellow) or *acquisition* (violet). This is visible in all views: The trajectory in the hidden state visualization moves along the border between the two corresponding areas of the different classes, the colors in the sequence switch between yellow and violet, and the heatmap visualization shows that both yellow and violet have higher confidence values compared to the other classes. Finally, Figure 5.7 (right) shows an example in which the confidence for each class remains similar while processing the sequence. The correct class has only a slightly larger confidence value than the other classes. However, the trajectory clearly shows that all corresponding hidden states are located near other hidden states of the same classification.

Incorrect Classification Figure 5.8 shows some examples of misclassifications. Figure 5.8 (left) provides an example similar to the previous one, where the confidence for each class is similar. The heatmap matrix shows that the model never distinguishes the *earns* class (the correct one) from the others, and eventually chooses an incorrect class at the end of the sequence processing. This indicates that these classes are difficult to identify using the model. In the projection, we can see that the trajectory is mostly located in the blue region (*money-fx* class). However, in Figure 5.6, we can also see that there are many sequences incorrectly classified as *money-fx* (blue). This means that many samples in this region belong to a different class. This is also visible in Figure 5.3, where we changed the colors in the projection to the actual classes of sequences instead of the classification results. At the top, where sequences were classified as *money-fx* (blue), the classes should actually be *crude-oil* (green) or *grain* (red) in many cases.

Figure 5.8 (right) shows an example in which the model is more certain toward a specific class (*acquisition* (violet)). However, the correct class is *earns* (yellow), which has a higher confidence only at the beginning of the sequence. When manually comparing different input sequences, it is sometimes also extremely difficult for humans to differentiate classes and correctly classify them. Multiple

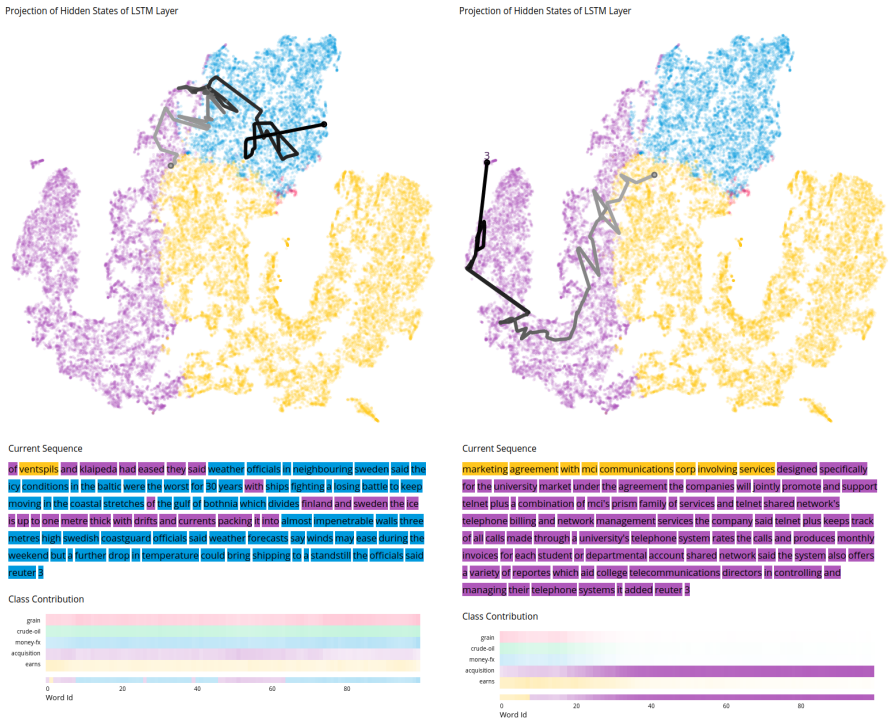


Figure 5.8 — Examples of incorrectly classified sequences in a model trained for multi-class classification. The heatmap matrix supports the identification of the classes for which the model becomes confused and on how the EP evolves throughout the training process, delivering an inaccurate result. In the left visualizations, the prediction is quite uncertain between all five classes. This is visible because the class contributions are similarly with low saturation for all classes. In the right visualizations, the model is rather certain in the classification of *acquisition* (violet), despite the correct class being *earns* (yellow). Underlying data source: Reuters [268].

classes are also occasionally appropriate. The fact that *earns* more often contains digits compared to *acquisition*, which more often contains continuous text, might be one possible explanation for the improper classification of this example.

5.1.5 Conclusion and Future Work

We introduced a visual analytics approach to address three interpretability challenges in the analysis of RNNs trained for NLP applications. We demonstrated through use cases some practical insights that one can achieve from using our techniques, and that can be instrumental in improving the interpretability of these models.

As a topic that has attracted strong interest in both the ML and visualization research fields, there are certainly more findings to come in the future. Notably, it would be interesting to see more research on interpretability issues from models addressing particular types of input data, such as videos [352], financial data [329], and eye tracking [191]. Interpretability is the main bottleneck faced by DL models, and more innovation in this topic will undoubtedly bring more improvements across many different applications. Additionally, an analysis of the visualizations generated with dimensionality reduction for the sequential data is important to assess whether the interpretation of the visualized time series is correct. Due to various interpretability problems of dimensionality reduction results (such as false or missing neighbors; see page 34 in Section 2.5.1), corresponding visualizations have the potential for misinterpretation. This topic is discussed in more detail in the next section, especially focusing on the challenges for temporal data.

5.2 Visual Quality of Time Series Projections

The previous section presented an approach where, among other techniques, dimensionality reduction was used to project multidimensional sequential data to 2D for visualization. Dimensionality reduction, in general, can be used to project time series data from multidimensional to 2D space and generate visual representations of the temporal evolution with the goal of revealing temporal patterns [45]. However, inevitable projection artifacts may lead to poor visualization quality and misinterpreting temporal information. The following visual analysis approach supports the interpretation of 2D projections of multidimensional sequential data.

This section is based on the following publications:

- T. Munz-Körner and D. Weiskopf. Exploring visual quality of multidimensional time series projections. *Visual Informatics*, 8(2):27–42, 2024, doi: 10.1016/j.visinf.2024.04.004 [22].
- T. Munz-Körner and D. Weiskopf. Supplemental material for "Exploring visual quality of multidimensional time series projections". DaRUS, V1, 2024, doi: 10.18419/darus-396 [21].

The images shown in this section were created with the following source code/material:

T. Munz, and D. Weiskopf. Visual analysis system to explore the visual quality of multidimensional time series projections. DaRUS, V1, 2024, doi: 10.18419/darus-3553 [20]. (The source code is also available on GitHub: https://github.com/MunzT/visualQuality.)

Visualizations for multidimensional temporal data generated with the result of dimensionality reduction (Section 2.5.1) can suffer from several interpretation issues (see page 34 in Section 2.5.1). Dimensionality reduction already has various interpretation problems in the traditional context of point set projection. For sequential data, projected samples are connected by lines or curves according to their sequential order to analyze the temporal evolution (see Figure 5.9 B). These visual connections may introduce additional artifacts and the danger of misinterpretation. Examples are related to the lengths of these connections and the interpretation of intersecting connections. With an uncertainty highlighting visualization (Figure 5.9 C), we emphasize potential regions for misinterpretation. An alternative halo visualization (Figure 5.9 D) enables users to concentrate on the reliable parts of the visualization by making uncertain parts less prominent.

This approach deals with RQ1, RQ2, and RQ4. Sequential data (RQ1) is represented as curves or connected lines using a projection to 2D generated with

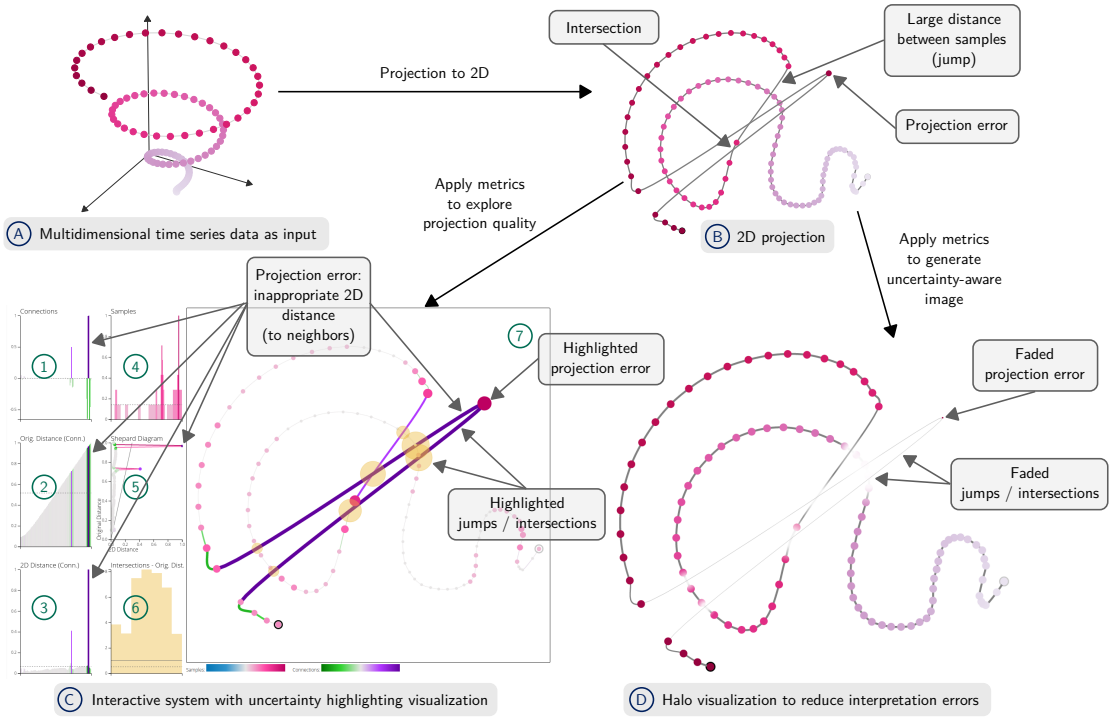


Figure 5.9 — Overview of our interactive approach to visualize and explore projections of multidimensional time series. (A) Multidimensional temporal data (a 3D spiral) is used as input. Samples are colored according to their temporal ordering from a brighter to a darker pink. (B) In a basic visualization of the projection (t-SNE), subsequent samples are represented as dots connected by curves. One time point (at the top right) was placed at a wrong location and could be incorrectly interpreted as a temporal outlier. Additionally, there are multiple intersections of connections and sudden strong changes (long connections) in distances between subsequent samples. Two visualizations can be used to deal with such artifacts: (C) By uncertainty highlighting (7), projection errors for samples and their connections are highlighted, and estimated distances for locations at intersecting connections are shown. Different quality metrics are mapped to the color (pink and blue for samples, purple and green for connections) and size of dots and in-between connections. Histograms show changes over time for time instances (4) and their connections (1), the distances between subsequent samples in the original space (2) and in 2D (3) along with their combination in a Shepard diagram (5), and the estimated distances at intersections (yellow) (6). (D) The halo visualization fades out artifacts and allows the focus on correctly projected data: the visibility is reduced according to the quality metrics, and halos mitigate the wrong impression from misleading intersections. The coloring of the samples shows the order of the data points.

dimensionality reduction. The visualizations provide interaction techniques to explore the visualizations themselves and link visual elements to other views of our system to explore properties of the temporal evolution of the projection. Parameter settings (RQ2) influence the size and appearance of visual elements and, hence, how well an error or correctly presented data is perceived. The most important setting of our approach is the choice of a quality metric; it decides how the visual elements are presented and how the visualizations are perceived. The use of dimensionality reduction on multidimensional sequences (RQ4) is central to this approach. Interpretation problems existing in visualizations created in such a way for temporal data are discussed, and new visualization techniques are proposed for better interpretation of such visualizations.

We summarize our contributions as follows: (1) We explore and characterize quality problems of projections for time series data introduced by lines or curves that connect time instances, and that may lead to misinterpretation. (2) We introduce uncertainty visualizations for multidimensional time series projections to reduce misinterpretation along the temporal path using two designs with different goals: an interactive visualization to explore projection artifacts and analyze problematic regions of the projection with uncertainty highlighting and a halo visualization where the representation of artifacts is adapted to better focus on well-represented data. (3) We adapt quality metrics for projections by considering the temporal relationships. (4) We provide a quality assessment at line intersections in 2D space by estimating the distance in the original multidimensional space. The source code of our approach is publicly available [20].

5.2.1 Related Work

In the following, we summarize relevant work for visualizing dynamic data, quality assessment of projection methods, and projection of multidimensional temporal data.

Dynamic Data

Dynamic data can be characterized as a sequence of successive observations. Since such data is almost omnipresent and highly relevant, there are many approaches to visualize them. Visualization methods range from line charts (for one-dimensional data) to stacked area charts [145, 147] (for multidimensional data). Spatial data can be plotted as connected scatterplots (e.g., a scanpath in eye tracking (see visualizations in Chapter 3)) on top of a 2D plane to show the temporal development. An more detailed overview of methods to visualize temporal data can also be found in Section 2.3. Typical for most of

these techniques is a clearly defined direction for the temporal component (e.g., from left to right), or observations are connected by line segments to show the sequential order. In the latter case, temporal information, such as speed, can be derived from the sampling rate and the spatial context or may get lost. Different visual encodings can be used to circumvent this [251].

The problem with all these techniques is that only a small number of dimensions can be visualized simultaneously. If data has hundreds or thousands of dimensions, they cannot be applied. In this case, methods to reduce the dimensionality are frequently employed.

Data Projection

Dimensionality reduction methods (see Section 2.5.1) are commonly used to handle multidimensional and high-dimensional data [115, 180] and provide the possibility to work with a reduced number of features. These techniques try to preserve important properties such as distances between samples or local neighborhoods. Van der Maaten et al. [320] provide an overview of many techniques. Typical examples include PCA, MDS, t-SNE, or UMAP (see Section 2.5.1). For linear methods [96] such as PCA and metric MDS, distances between all samples can be compared globally; nonlinear methods [203] such as non-metric MDS, t-SNE, and UMAP try to show both local and global structures.

The results of projection methods are frequently used to visualize multidimensional data [115, 180]. However, a lot of information may get lost and it is often impossible to faithfully represent high-dimensional data in a lower-dimensional space [180]. Especially in a very low-dimensional space such as 2D, errors in the projection can often not be avoided. Therefore, there are fundamental problems of projection errors and interpretability.

A survey of different visual analytics approaches for dimensionality reduction is provided by Nonato and Aupetit [236], who also include an overview of techniques for evaluating distortions. For assessing the quality of a projection, different quality metrics were introduced; overviews are available by Espadoto et al. [118] and Gracia et al. [134]. The term *metric* is used to refer to a broad range of methods for a quantification of quality characteristics, and not all of these measurements strictly adhere to the mathematical definition of a metric. This understanding of the term is also reflected in the publications mentioned.

There exist visualization approaches to show projection errors such as distortions/tears and artifacts. Many approaches use color coding to indicate areas with poor projection results in a point-based fashion [229], on a map as a

static image [283] or as an interactive system with the possibility to explore the quality in relation to a reference point [43]. Interactive systems also allow exploring the quality in multiple views with different visualizations to guide the exploration [123]. Some techniques are specifically designed for one projection method, such as t-viSNE [83]; others can be used more broadly (e.g., the method by Liu et al. [208]). Aupetit [43] proposes an interactive proximity-based approach where projection errors are shown relative to a specified sample using a colored map generated from Voronoi cells. ProxiViz [152] is based on this approach and provides improved interaction techniques. Stress Maps [283] show local stress on a map by calculating point-wise stress values to highlight false neighbors. CheckViz [204] visualizes tears and false neighbors on a map using Voronoi cells. Martins et al. [219] propose an interactive system that allows the analysis of the neighborhood of reference points to find missing and false neighbors. Liu et al. [208] visualize point-wise distance measurements in an interactive system that also allows modification of the data through movement and deletion. Rieck and Leitte [270] use persistent homology to compare the quality of dimensionality reduction methods. Stahnke et al. [298] explore the concept of probing and show where a better position for projected samples could be, and Raj and Whitaker [262] use order-aware projection to improve interpretability. DimReader [122] is an approach for reconstructing the coordinate axis for nonlinear projection techniques to study infinitesimal perturbations.

These approaches can be used on multidimensional data, but they were not designed for quality assessment of dynamic data. They do not consider the additional component of time and the relationship to temporally adjacent samples.

Projecting Dynamic Data

Different projection methods have been applied to dynamic datasets from various application domains to generate visualizations or interactive systems to explore the temporal evolution. These methods have in common that they use a dimensionality reduction technique on multidimensional temporal samples to project them to 2D for visualization. Often, the linear projection methods PCA and classical MDS [312] (both create the same projections [320]) are used (e.g., Brown et al. [72]). Also nonlinear methods are frequently employed, including non-metric MDS (e.g., van den Elzen et al. [318]), as well as t-SNE (e.g., Garcia et al. [5]) and UMAP (e.g., Ali et al. [35]). These approaches connect subsequent samples by either straight lines (e.g., van den Elzen et al. [318]) or smooth curves (e.g., Bach et al. [45]) in order to show the temporal connectivity. An example of the connection by smoother curves is the use of Catmull-Rom

splines [45]. Some approaches use sliding window approaches to generate high-dimensional data (e.g., Ali et al. [35]) or a smoother result (e.g., Ward and Guo [333]). It can be seen that they all use the same basic concept with some variations. This general approach is used in many different application areas such as text [216], simulation [230], videos [45], graphs [318], sensor data [75], medicine [68] and many others. It is apparent that there is a high interest in using dimensionality reduction for the visualization of dynamic data in different domains. The system introduced in Section 5.1 also represents such an approach. An overview of publications using projection methods on multidimensional temporal data for visualization is visible in Table 5.1. All of these approaches use visualization of projected data to explore the temporal evolution, but none explore whether we can trust the generated visualizations, and they often ignore the fact that some observations could result from projection errors. Especially the publications about Time curves [45] and Projection Path Explorer [155] discuss how projections for temporal data can be interpreted. However, they do not mention that there might be problems with the interpretation caused by projection errors or the visualization itself. Brown et al. [72] mention the problem with projection errors, but they do not deal with it since it is out of the scope of their work.

Therefore, we aim to arrive at quality measurements that describe the uncertainty of the visual mapping according to the projection quality and use them in adapting strategies from uncertainty visualization [69, 214, 245, 256].

An evaluation of the quality of projections of dynamic data is presented by Vernier et al. [324]. However, their quality assessment is only performed globally for the full projection. There are further approaches that deal with the projection of dynamic data. For instance, MultiProjector [98] is an approach where a different representation of time was chosen: time is plotted on a third dimension to mitigate overplotting issues of dimensionality reduction. Dynamic t-SNE [264] generates a sequence of images where, for each time step, one projection is constructed, and each point represents one dataset. In our work, we do not consider such approaches since we focus only on static explorable visualizations.

5.2.2 Uncertainty Modeling

When projecting multidimensional data to 2D, not all of the important information can be maintained. Therefore, it is also possible to obtain the same 2D representation for different multidimensional structures. For temporal data,

Table 5.1 — Publications using dimensionality reduction on multidimensional time series to project them to 2D for visualization and exploration. We provide information about the input data, projection method, connection type, and some further properties for each approach.

Year	Publication	Method Name	Input Data	Projection	Connections	Remarks
2007	Mao et al. [216]	–	Sequential text documents	PCA, MDS	Smooth curves	Visualization of multiple documents
2007	Schreck et al. [278]	–	Financial data	SOMs [184]	Straight lines	Trajectory bundle visualization
2010	Hu et al. [162]	Motion Track	Human motion sequences	SOM [184] and LLE [274]	Smooth curves	Glyphs show images of motion, multiple motion sequences
2011	Ward and Guo [333]	Shape Space Projections	Stock market data	PCA	Straight lines	Long time series, sliding window approach
2012	Bernard et al. [57]	TimeSeriesPath	Earth observations	PCA	Straight lines	Data aggregation, interactive analysis
2012	Chen et al. [85]	Hierarchical Parametric Histogram Curve	Sequential text documents	Distributional scaling [260] (variant of MDS)	Straight lines	Pictures at curve points
2014	Molchanov and Linsen [230]	–	Ensembles data for climate and star evolution	PCA, MDS	Straight lines	Multiple time series: simulations with different parameters
2015	Bach et al. [45]	Time curves	Data-agnostic: video frames, Wikipedia article histories, brain connectivity	Classical MDS	Smooth curves	Sample size/color and line thickness/color encode additional information (e.g., time), geometric characteristics and patterns are described
2016	Van den Elzen et al. [318]	–	Dynamic networks	PCA, non-metric MDS, t-SNE	Straight lines	Smoothing the data with a sliding window, color of points according to time or some attribute value
2016	Zhu and Chen [356]	Performance Histogram Curves	NBA (basketball) games	LMDS [101] (variant of MDS)	Straight lines	Sliding window approach for discretization, Gaussian kernel for smoothing kernel
2017	He and Chen [146]	–	Representation learning	Temporal t-SNE	Smooth curves	Multiple time series
2017	Rauber et al. [265]	–	Activations in NNs	t-SNE	Straight lines	Multiple time series
2018	Brown et al. [72]	ModelSpace	User-interaction behavior	Non-metric MDS	Straight lines	Multiple time series, sample size/color and line thickness/color encode additional information
2018	Cakmak et al. [75]	ViCCEx (Visual Chemical Contamination Explorer) system	Sensor data	t-SNE	Straight lines	Multiple time series
2019	Ali et al. [35]	TimeCluster	Real-world data from medicine and biology	PCA, t-SNE, UMAP, DCAE [163]	Straight lines	Sliding window approach to generate data, long time series, low-dimensional data
2019/ 2021	Steinparz et al. [299], Hinterreiter et al. [155]	Projection Path Explorer	Decisions made by humans and machines	PCA, t-SNE, UMAP, Isomap [307]	Smooth curves	Multiple time series, describes patterns emerging from multiple trajectories
2020/ 2022	Brich et al. [67, 68]	–	Medical surveillance data	PCA, metric MDS, UMAP	Smooth curves	Multiple time series
2020/ 2021	Garcia and Weiskopf [129], Garcia et al. [5]	–	Hidden states of LSTMs	t-SNE	Straight lines	Colored by classification result; see Section 5.1
2020/ 2022	Hägele et al. [142], Hägele et al. [143]	Robot path evolution	Robot motion planning	PCA	Straight lines	Smoothing: aggregating several time steps with a sliding window, multiple time series
2021	Tsutsumi et al. [313]	Reaction space projector (ReSPer)	Chemical reaction mechanisms	Classical MDS	Straight lines	Multiple time series
2022	Toda et al. [309]	EvoMapper	Evolutionary computation	t-SNE	Straight lines	Multiple time series
2023	Bauer et al. [52]	–	Ensemble data of liquid and gas flow in porous media	Non-metric MDS, t-SNE, UMAP	Straight lines	Multiple time series
2023	Tarner and Beck [306]	Runtime Evolution Paths	Runtime data of software systems	UMAP, Aligned-UMAP	Straight lines	Multiple time series

this means that we can get the same image for different time series. Often, there exists no best projection method to represent the original data and show the most important features without projection errors and additional artifacts created by the visualization. Our goal is to make the visualizations aware of the uncertainties introduced by the projection and to minimize possibilities that may lead to misinterpretation.

Characterization of Misinterpretations

Projection errors already occur in any traditional point-based projection (see Section 2.5.1). However, in the visualization of temporal data, more information is available than in classical projections. New for temporal data are the connections between samples to show temporal connectivity. These connections are additional sources for misinterpretation.

Local Effects The length of the line that connects two subsequent data samples is intuitively interpreted as a distance. For our plots of temporal data, we naturally relate it to the speed (of changes in the data) along the time series: a large distance corresponds to a high speed of changes in data values. Therefore, large variability in distances between samples would indicate that there is a sudden or strong change in data values. Since these lengths are determined by the location of projected subsequent samples, they highly depend on the chosen projection method and are influenced by local stress between adjacent samples. Therefore, this specific type of local stress plays a highly relevant role in time series projections and makes it different from general point cloud projection. We know that the type of projection method (linear or nonlinear) has a strong influence on varying distances, and this might even be a property of the projection method when creating clusters for neighboring samples. Nevertheless, this might lead to interpretation problems, especially for less experienced users. From these considerations, we arrive at a characterization of misinterpretation scenarios as illustrated in Figure 5.10:

C1 Varying distances between samples are linked to varying speed, which will be misinterpreted if the varying distances do not appear in the original data.

C2 An extreme case of the above is an erroneous "jump." We use this term for a large distance between two samples while other samples are relatively nearby. This term was already introduced in Section 5.1.3 on page 155.

C3 An incorrectly projected sample could be misinterpreted as an outlier. This is a special case of C2 with two jumps: one before and one after a

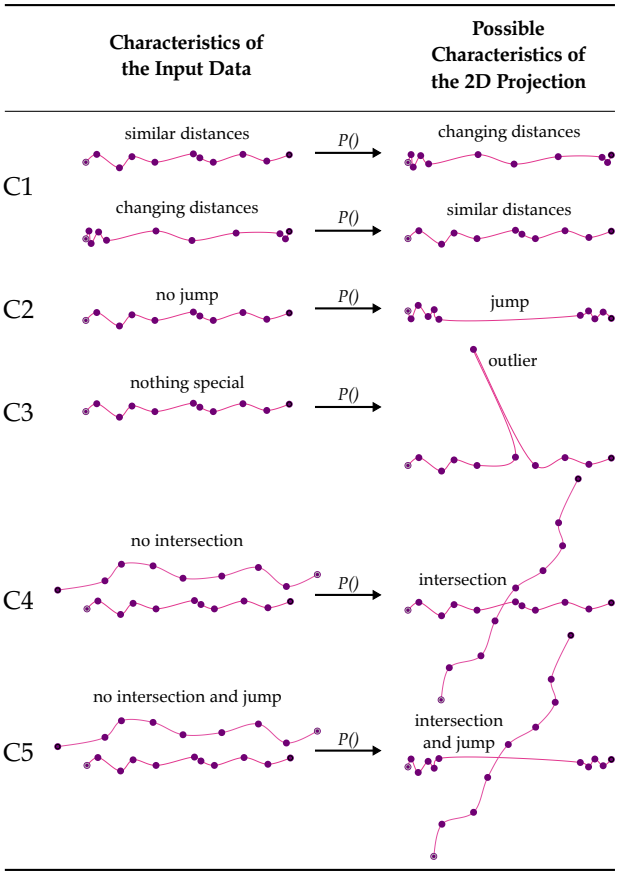


Figure 5.10 — Characterization of projection errors that may lead to misinterpretation. The left side shows simplified properties of the high-dimensional input data regarding distances between samples and intersections of their connections. The right side shows possible 2D projections (created via the projection $P()$) that do not match the properties of the original data.

sample. However, the interpretation of an outlier is quite different from that of a jump.

Global Effects Intersections are caused by drawing connections between samples. They are a global effect because intersections can occur for connections that are very far apart temporally along the time series, which makes this problem structurally different from the above local effects. Another difference is that the intersection position does not directly result from projecting any original data point. In fact, the intersection in 2D depends on the location of the pairs of adjacent points but also on how connections between points are drawn (e.g., straight lines vs. curves). Therefore, intersections result from a complex interplay of the projection and the visualization of connections.

The natural interpretation of intersections is the recurrence of a temporal state. However, it is very likely to have intersections in 2D space that do not correspond to intersections in the original multidimensional space—or that do not even correspond to close-by transitions along the time series. It is a fundamental and intrinsic problem: due to the curse of dimensionality [107], high-

dimensional space becomes vast and largely empty, with very little probability of hitting another part of the time series by chance. In contrast, 2D visualization space is very restricted and greatly enhances the probability of intersections. Wrong intersections can be summarized in the following misinterpretation scenarios (Figure 5.10):

C4 There could be an intersection in the 2D projection, but none in the data.

C5 An incorrect intersection often comes with a jump (see C2) because a jump covers a large distance and, thus, increases the chances of an intersection.

The five scenarios (C1–C5) are prototypical for misinterpretation issues. There can be variants and combinations of these, and there might be other additional scenarios (e.g., an intersection in the original data that was not projected as an intersection) that are less likely to happen.

Relationship to Established Visual Patterns Bach et al. [45] describe several visual characteristics and patterns in projections of temporal data. Many of them are directly related to the interpretation issues we have outlined above. Their temporal (or curvilinear) distance is identical to our discussion of the length of connections. Related characteristics are their "point density" (density of samples along the curve), "transitions" (as our "jumps"), and "outliers" (identical to our definition). Therefore, these patterns in the visualization can be directly mapped to our misinterpretation scenarios from local effects. Other patterns from Bach et al. correspond to misinterpretations from global effects: their "self-intersections" and "cycle" patterns. These can be directly mapped to intersections from our description.

These visual patterns are easy to perceive and lead to an intuitive interpretation of the underlying data. They are very strong visual features in the projection, especially caused by the easy-to-perceive connecting curves. This is a great advantage for reading such visualizations and using it for communication. However, this advantage comes with the equally problematic disadvantage if the visual patterns do not correspond to the input data. Therefore, time series projections come with the serious danger of massive misinterpretation. With our uncertainty modeling and visualization, we want to address this issue by making the visualization aware of inaccuracies and communicating them clearly to the user.

Requirements

We formalize requirements for our visualization approach to reduce misinterpretation of the visual elements as follows:

R1 A visualization that supports users in identifying projection artifacts concerning samples, lines, and intersections.

R2 Exploration of visual elements (samples, lines, and intersection), especially their relationship to their temporal and local neighborhood through interactive visualizations.

R3 Quality metrics that consider the temporal relationships of samples and/or their connections; these metrics should be integrated in the visualizations.

R4 A quality assessment at line intersections in 2D space to better understand if an intersection is related to low proximity or the result of a projection error/artifact from the visualization.

R5 A visualization that supports users in focusing on well-represented data.

These requirements are based on several sources: First, we derive them from the characterization of issues from misinterpretation (see above). Second, we also base the requirements on our experience with dimensionality reduction and XAI, including supervised and unsupervised ML. More details about specifying the requirements is provided in Section 2.6.

Processing Pipeline

In our approach, it is possible to analyze just one or multiple time series with the same number of dimensions for each time step. For the following pipeline steps, we treat all samples of each time series as one set. First, we may use standardization (z-score normalization) for all our features. This step is usually necessary if the different features come from different measurement scales. However, if values have a low standard deviation, this step could also negatively impact (e.g., noise that gets too much importance) and be neglected [203]. Afterward, if the input data's dimensionality is very high, all samples get projected to 50 dimensions using PCA. Lee and Verleysen [203] recommend this step before applying a nonlinear dimensionality reduction technique because it removes many meaningless dimensions from very high-dimensional data before continuing with the next steps. Finally, all samples are projected to 2D using the actual linear or nonlinear projection method to obtain coordinates for visualization.

Next, we determine intersections in 2D for the lines or curves used as connections in the visualization. In our approach, we provide both of these connection types since previous work showed that both techniques are equally used (e.g., Bach et al. [45], van den Elzen et al. [318]), usually without justification why specifically one of them was chosen. Moreover, we perform interpolation in multidimensional space to get estimated data at intersections (R4). Then, we calculate quality metrics for samples (R3), lines (R3), and intersections (R4). Finally,

in an interactive visualization, samples, lines, and intersections can be explored by brushing and linking to explore the quality metrics' temporal development and locations of artifacts (R1–R2). Alternatively, the halo visualization can be used where projection and visualization artifacts are faded out (R5).

Multidimensional Dynamic Data Model

Our input data D consists of m time series T_g, each containing N_g n-dimensional samples (also referenced as time steps, instances, or observations). In other words, we have multiple multidimensional time series with possibly different lengths, but the dimensions are the same for each instance of each time series.

The h-th sample of time series T_g is defined as n-dimensional vector: $\boldsymbol{x}_{g,h} = [x^1_{g,h}, x^2_{g,h}, \dots x^n_{g,h}]$ with $x^l_{g,h} \in \mathbb{R}$, $1 \leq l \leq n$, $n \geq 1$ and $1 \leq h \leq N_g$. A time series T_g consists of N_g samples: $T_g = [\boldsymbol{x}_{g,1}, \boldsymbol{x}_{g,2}, \dots \boldsymbol{x}_{g,N_g}]$.

Now, we use the concatenation of all samples of all m time series. The overall number of samples is $\overline{N} = \sum_{j=1}^{m} N_g$, and we have a list of all data samples $\overline{D} = [\boldsymbol{x}_{1,1}, \dots \boldsymbol{x}_{1,N_1}, \dots \boldsymbol{x}_{m,1}, \dots \boldsymbol{x}_{m,N_m}] = [\boldsymbol{x}_1, \dots \boldsymbol{x}_{\overline{N}}]$.

Pairwise Euclidean distances between two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are determined as $d_{i,j} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$ with $\boldsymbol{x}_i, \boldsymbol{x}_j \in \overline{D}$. The distances in image space, after applying a projection method $P()$ to reduce the original space to two dimensions, are determined as $\delta_{i,j} = ||P(\boldsymbol{x}_i) - P(\boldsymbol{x}_j)||$. For the distance between subsequent samples at time t and $t+1$, we set $i = t$ and $j = t+1$. Here, $\boldsymbol{x}_t, \boldsymbol{x}_{t+1} \in T_g$ belong to the same time series.

In our calculations, we use Euclidean distance, but this could be replaced by other distance metrics. We chose Euclidean distance because it is widely used and fits MDS or PCA projections. A different metric might fit better for other projection methods. However, it is difficult to decide which distance measurement is best to use. This depends on both the structure of the input data (unknown before analysis) and the projection method. Future work could investigate using other metrics.

Projection Quality

It is known that different artifacts can be created by dimensionality reduction, such as false and missing neighbors. This leads to points projected to wrong positions and variability in distances between neighbors. The position of these points influences their connections that may lead to misinterpretation; this is already described on page 172. Additionally, the points themselves may also

contribute to misinterpretation, and their projection quality should be considered within the temporal context. Different quality metrics are available for the quality assessment of dimensionality reduction methods [118, 134]. Vernier et al. [324] even investigated how to adjust them for a global measurement of temporal data. In contrast, we need local quality metrics to indicate the visual quality of the projection for time series (R3).

Pairwise Metrics As the first type of quality metric, we introduce temporal quality assessment for connections that are computed by only considering distances between temporally neighboring samples instead of all samples. This means that we use only distances between $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$. More precisely, for an error estimation along connections, we calculate *sequential distances* of connections in the multidimensional space and 2D:

$$q_{c1}(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) = \delta_{t,t+1} - d_{t,t+1} \qquad q_{c1} \in [-1, 1]$$

All distances have to be normalized such that they can be compared between the original space and 2D, and therefore, it is $\delta_{t,t+1} \in [0, 1]$ and $d_{t,t+1} \in [0, 1]$. This quality measurement shows the error in 2D when the goal was to get the same distances represented as in the original space. Negative values toward -1 indicate that a distance is too small, and values toward $+1$ that the distance is too large. It is possible, and sometimes useful, to scale the resulting values, e.g., using standardization. This allows that smaller quality variances can be better explored, e.g., if the maximum errors are not very large.

Alternatively, we provide a *sliding window approach* that extends the above measurement to include variability in a larger temporal neighborhood. We calculate for each connection in the sliding window of odd size w the normalized difference between the distances in the original space and 2D and compare the value of the actual connection to the mean value of the others. For an odd w that considers the same number of connections before and after the actual connection, we use:

$$q_{c2}(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) = \left(\delta_{t,t+1} - d_{t,t+1} - \frac{\sum_{i=t-\lfloor w/2 \rfloor, i \neq t}^{t+\lfloor w/2 \rfloor} (\delta_{i,i+1} - d_{i,i+1})}{w-1} \right) /2$$

$$q_{c2} \in [-1, 1]$$

This quality metric generates a smoother result compared to the previous measurement by also considering a local temporal neighborhood and determining a difference to the local mean distance.

Point-Oriented Metrics The second class of quality metrics uses a traditional approach for local projection quality in general. They consider data samples individually and take into account the local neighborhood without using information about their temporal relationships. For the first measurement, we consider a local neighborhood of size k to calculate the differences between normalized distances in 2D and multidimensional space (*local neighborhood distances*):

$$q_{s1}(\boldsymbol{x}_t) = \frac{\sum_{i \in U_{t_k}} (\delta_{t,i} - d_{t,i})}{k} \qquad q_{s1} \in [-1, 1]$$

Here, we define U_{i_k} as the local neighborhood of $\boldsymbol{x}_i$ in the original space, consisting of k samples. If samples belonging to the local neighborhood are far away, the value gets larger. A value toward -1 means that neighboring samples are too close and a value toward $+1$ that they are too far away.

Additionally, we can determine the number of samples that are both within the local neighborhood of $\boldsymbol{x}_t$ in 2D and the original space. V_{t_k} is the set of k nearest neighbors of $\boldsymbol{x}_t$ in 2D and U_{t_k} in the original space. We determine the *local neighborhood preservation* as:

$$q_{s2}(\boldsymbol{x}_t) = 1 - \frac{|V_{t_k} \cap U_{t_k}|}{k} \qquad q_{s2} \in [0, 1]$$

Here, 1 means that the neighborhood in 2D and the original space are completely different and 0 that the neighborhood is the same.

In our visualization approach, we provide access to these metrics in order to facilitate different perspectives on quality. In all our visualizations, we use $k = 5$ and $w = 7$. We also tested additional approaches reflecting quality aspects of a projection. Some of them can also be used in our system to highlight alternative distance or neighborhood-related aspects of projections. Therefore, we adapted existing approaches (Espadoto et al. [118], Gracia et al. [134]), usually used for global assessment, on a local set of samples or distances. For samples, this includes sliding window or local neighborhood approaches based on normalized stress, correlation, trustworthiness, continuity, root mean squared error (RMSE), and others. For the connections, we implemented normalized stress using a sliding window. In our experiments, we noticed that our previously presented methods created very appealing but also useful results. However, depending on which aspect of projection quality someone wants to explore, one of these alternative methods can be used instead. The quality measurements can be switched interactively in our visualization system.

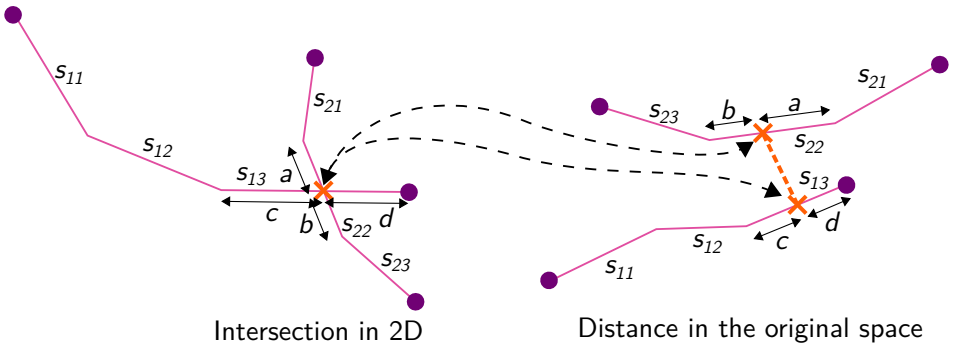


Figure 5.11 — Illustration of our approach for determining distances in the original space for intersections in 2D. Here, Catmull-Rom splines constructed from three line segments are shown. Left: An example in 2D for two curves with one intersection. The intersection occurs for the line segments s_{13} and s_{22}. Curve parameters a–d are associated with the length of the line segments where the intersection occurs. Right: Simplified illustration for the original space. In the corresponding line segments (s_{13} and s_{22} in the original space) of the multidimensional curves, the same curve parameters are used to determine corresponding positions on the two involved multidimensional line segments. Between these multidimensional interpolated positions, a distance can be determined.

Intersection Reconstruction and Quality

We aim to understand if intersections of connections between points in 2D correspond to intersections in multidimensional space or not (R4). To this end, we have to first determine the position of line or curve intersections in 2D. And then, we reconstruct the corresponding two points in multidimensional space: one point on each of the connections. It is important to note that these intersection points are not in the set of original data samples. Therefore, we need a mechanism to reconstruct them. Finally, we can compute the distance of the two points in multidimensional space. This distance is an indicator of the error introduced by the intersection in 2D: the larger the original distance, the larger the visualization error.

We support both straight lines and curves as connecting elements. Curves are modeled as centripetal Catmull-Rom splines [353]. For the calculation of intersections between two lines, we use the Bentley–Ottmann sweep-line algorithm [56]. The intersection computation in 2D is straightforward for straight lines. For Catmull-Rom splines, we use the individual line segments to determine the intersection based on segments; in Figure 5.11, for example, from line segments s_{11}, s_{12}, and s_{13}.

To reconstruct corresponding points in multidimensional space, we use piecewise linear interpolation there. For straight lines, we readily interpolate in multidimensional space. For Catmull-Rom splines, we generate a multidimensional Catmull-Rom spline first. Then, we find possible positions of the representative multidimensional samples on multidimensional connections by linear interpolation of the corresponding line segments of the multidimensional curve.

In Figure 5.11, there is an intersection between the line segments s_{13} and s_{22} in 2D. The reconstruction is performed on the corresponding segments (s_{13} and s_{22}) in the original space. While this approach should return an accurate estimate of the interpolated point in the original space when using linear projection, the result may be wrong for nonlinear projection methods due to the nonlinear behavior between samples. However, for small enough segment lengths, the approximation leads to good accuracy. The numerical quality could be further improved by applying curve subdivision and bisection to determine the multidimensional points. Finally, a metric value for intersections is determined by calculating the distance between the two estimated positions of samples in the original space (see Figure 5.11).

5.2.3 Visual Analysis Approach

Our goal is to provide an exploration system for projections of multidimensional time series data to minimize misinterpretation and to be aware of uncertainty (R1, R2, R5). To achieve this goal, we use the aforementioned metrics (R3, R4) to assess the quality of the projection. Time is not part of the projection and is only implicitly added by connecting subsequent samples. We draw projected samples as dots and connect them by either lines or curves to show the temporal sequence. In case we connect them by curves, we generate a piecewise linear centripetal Catmull-Rom curve [353]. This category of Catmull-Rom curves is aesthetic and does not self-intersect. In our system, it is possible to show multiple time series together to compare temporal behaviors.

We provide two different visualization modes with different design goals. The first one (Figure 5.12) highlights areas with wrongly projected areas (R1) and allows interactive analysis of projection errors (R2) according to different quality metrics (R3). With interactive exploration, users can identify projection errors (R1) and verify for intersections what they mean (R4) for the actual input data. The second one (Figure 5.13) introduces visual changes to support users in focusing on correctly projected data (R5) without being distracted by projection errors and misleading representations.

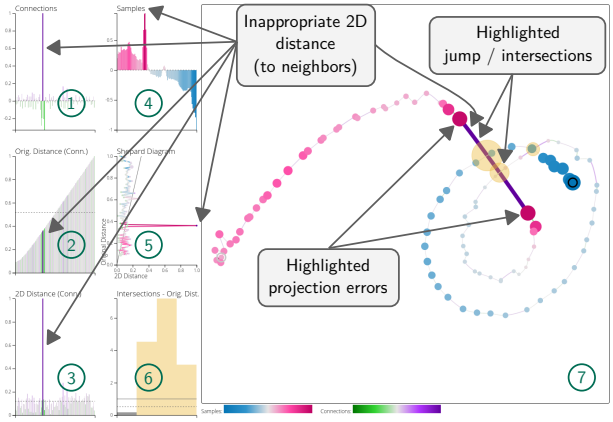


Figure 5.12 — Screenshot of our interactive uncertainty visualization (7) with data plots (1–6) for temporal analysis: temporal evolution for the metrics of time instances (4) and their connections (1), distances between subsequent samples in the original space (2), in 2D (3), and as a combination in a Shepard diagram (5), and estimated distances at intersections (6). The same spiral data as in Figure 5.9 is used; UMAP was applied for the projection.

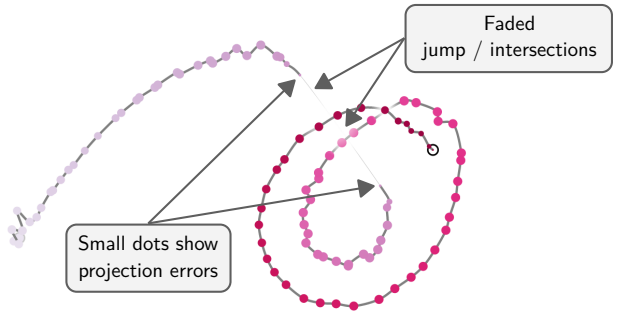


Figure 5.13 — Halo visualization of the same data and projection method used in Figure 5.12.

The visualizations contain dots and connections using different colors and sizes that correspond to quality metric values. We show highlights (Figure 5.12) or halos (Figure 5.13) for intersecting lines in 2D whose size depends on the distance in multidimensional space. Our approach can be applied to any dimensionality reduction technique that can be used to create 2D projections. Even for nonlinear methods, it supports the awareness of different distance-related aspects.

Uncertainty Highlighting and Interactive Exploration

Our interactive analysis systems (Figure 5.12) allows the exploration (R2) of different quality metrics (R3) using brushing and linking [332] on different views [273]. The visualization of the projected data is visible in the center of the main view (Figure 5.12 (7)), showing quality properties for samples, connections, and intersections (R1, R3). Here, different quality metrics are used for visual encoding in the form of color and size for data samples and connecting lines. The temporal evolution of the metric values for samples and their connections

are also shown in histograms on the left side to better evaluate changes and spot outliers in the temporal context (Figure 5.12 (1, 4)). Similarly, both 2D and multidimensional distances between neighboring samples are visible along the timeline (Figure 5.12 (2, 3)) as reference. Their relationship is also shown as Shepard diagram [176] (Figure 5.12 (5)). Usually, in such a diagram, the pairwise relationship between all samples is visualized; we use it only on the subset of temporally neighboring samples and add lines between each observation to show the temporal development. Finally, the estimated multidimensional distances at intersections (Figure 5.12 (6)) can be inspected as well. In all these plots, except for the Shepard diagram, time is plotted on the x axis and the metric value characterizing the quality or distance along the y axis. In the Shepard diagram, the 2D distances are mapped to the x axis and the original distances to the y axis. For the quality metric plots (Figure 5.12 (1) and (4)) and the distance plots (Figure 5.12 (2) and (3)), the mean values are shown as dotted lines. In Figure 5.12 (6), where distances at intersections are shown, the same mean value as in Figure 5.12 (2) is depicted that was computed for all connections in the original space. Additionally, a solid line in Figure 5.12 (6) indicates the maximum length of all connections. In the Shepard diagram, a diagonal line shows where points should ideally be located to have a good projection that preserves distances proportionally.

Hovering a line, dot, or intersection in either the view with the projection (Figure 5.12 (7)) or the diagrams (Figure 5.12 (1–6)) on the left highlights the corresponding elements also in the other views. This helps in the exploration of both temporal and local relationships. While it is possible to follow in each view the temporal path, the projection allows the exploration of the neighborhood additionally, and in the other views, temporal relationships might get better visible. A tooltip can be shown containing additional information about the selected element.

Quality Metrics and Visual Encoding

We apply the previously described quality metrics (page 176 in Section 5.2.2) to get both point-based and pairwise values to encode samples and connections (R3). Usually, a good quality value equals 0, and values indicating an artifact go toward 1 or -1 (usually, positive values mean a value is too large, and negative ones mean too small, depending on the chosen metric). Additionally, especially when only small variances are available, standardization can be used to stretch the value range in such a way that the mean value represents 0, and the whole color gradient is used to better explore smaller variances or distributions with few outliers.

We use the values of quality metrics for the color and size of lines, dots, and semitransparent disks (highlights) at the position of intersections. Absolute values are used to determine both color brightness and element size. Lower values for dots and lines get a brighter color than high values. For samples, we use pink for positive values and blue for negative ones; for connections, we use purple and green, respectively. As we usually expect the mean value as "perfect" distance, we assign a value of 0 (using standardization) and draw corresponding dots in gray. Line thickness and dot size are larger for higher absolute values. Note that both color value and size are possible visual variables that can be used for uncertainty encoding [214].

The actual metric values for distances can be explored using tooltips and brushing and linking to interact with additional plots (Figure 5.12 (1–6)). As an example, Figure 5.12 (1–3), (5), and (7) show color information about connections. Green indicates a too small metric value, and purple means a too large one. Often, green and purple connections are located temporally next to each other when using a sliding window approach. For samples, Figure 5.12 (4), (5), and (7) contain information; pink indicates a too large metric value and blue a too small one. For example, when using the quality metric q_{s1} (local neighborhood distances) for samples, pink and blue dots indicate that distances to neighboring samples are too large or too low, respectively. When using q_{c1} (sliding window approach) for connections, purple and green connections mean that the distance is too large or too low, respectively, according to their temporal neighborhood. Besides the previously mentioned quality metrics, it is also possible to color samples and connections by time or use the distance between neighboring samples in the original space, reflecting the speed between samples if they were temporally equidistant. Since a large number of time instances makes it sometimes difficult to explore both dots and lines due to overplotting, elements can be hidden or drawn in a neutral subtle way to focus on only one type of visual elements.

We draw highlights for intersections in gray when their represented distance is smaller than the mean distance of every distance between two subsequent samples. If the distance is larger, yellow disks are used instead. We added this differentiation since we wanted to categorize the type of distance as well. The size of the disk is scaled depending on the maximum distance estimated at all intersections to avoid visualization of very large disks. These highlights are plotted on top of the other elements and are slightly transparent in order to be able to see the projected data and areas with multiple highlights. The colors are used both in the plots on the left side and in the projection.

Halo Visualization

The previously described artifact highlighting draws much attention to parts with low projection quality (R1) and, thus, it may impair the visualization of the other, reliable parts of the visualization (R5). This problem is addressed by a complementing uncertainty-aware visualization that uses halo rendering (Figure 5.13). The idea is that samples at wrong locations in 2D and respective connections should be drawn small and thin and intersections in 2D that are not represented in the original data should be faded out to prevent false interpretations. Therefore, the halo visualization does not highlight artifacts but rather reduces the visual perception of these elements.

An example is shown in Figure 5.13: While a visualization with connected points suggests a large jump with two intersections in our exploration system (Figure 5.12), corresponding samples in our halo visualization are small compared to other samples, and the connecting line is very thin. Additionally, the line with intersections is discontinued to show that no intersection is available in the original data.

We achieve this inverse impression by reversing the contribution of metric values of the quality to the size of samples and lines and putting a white halo under the intersecting lines similarly to the haloed line effect used in 3D visualizations [42, 119]. The halos are more blurred on the outside; relying on fuzziness is an intuitive indication of uncertainty [214]. Color is used to represent the temporal ordering of samples.

Implementation

The data preparation and preprocessing steps of our approach are performed with scripts implemented in Python 3 [321]. Projections are performed with Scikit-learn [250] for PCA, metric MDS based on SMACOF [100], and t-SNE; additionally, we use the UMAP library [225]. Our interactive system has a web interface and is implemented with JavaScript and Python 3; visualizations are generated with the D3.js [65] library.

5.2.4 Examples

In the following, we explore results generated with our approach for different types of datasets. First, we apply it to artificially generated data that contains typical temporal patterns. Then, we use simulation and a real dataset. In all our experiments, we performed feature standardization in preprocessing. In all images, we used Catmull-Rom curves for connections.

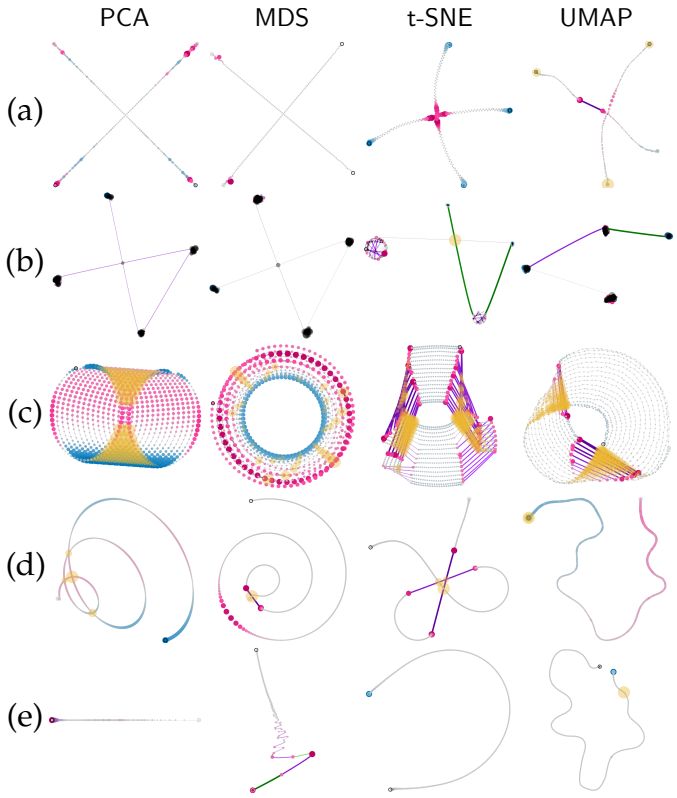


Figure 5.14 — Different examples where artifacts are highlighted for PCA, MDS, t-SNE, and UMAP projections. From top to bottom: (a) two intersecting lines, (b) multiple clusters connected by intersecting jumps, (c) 3D spiral, (d) 3D spiral with varying radius, and (e) variance in the distance between samples.

Artificially Generated Data

We first demonstrate our approach on artificially generated data. The data has a relatively low number of dimensions to explore the influence of our approach to data we can still intuitively understand. We show examples for scenarios C1–C5. Our examples use 3D data for circles, spirals, clusters, and intersecting lines and consist of 200 time steps for Figure 5.14 (a), 100 for Figure 5.14 (b), and 1000 for Figure 5.14 (c)–(e). The number of intersections varies between 0 and 358 depending on the data and the projection. Since our approach is projection-agnostic, we applied different methods frequently used by the visualization community (PCA [160], MDS [187], t-SNE [319], and UMAP [225]) as already introduced in Section 2.5.1. For samples, we use the quality metric q_{s1} (local neighborhood distances), and for connections q_{c2} (sliding window approach), as described on page 176 in Section 5.2.2.

As common artifacts, we observed jumps in the 2D projection that are not present in the original data (C2, C5). Figure 5.14 (d) shows for a 3D spiral examples of MDS and t-SNE, and Figure 5.14 (c) for a 3D spiral with increasing radius of t-SNE and UMAP. In these cases, the distance in the original space is small, and in 2D large. The connections are, therefore, rather thick and purple. Additionally, samples before and after a jump are highlighted in pink since the

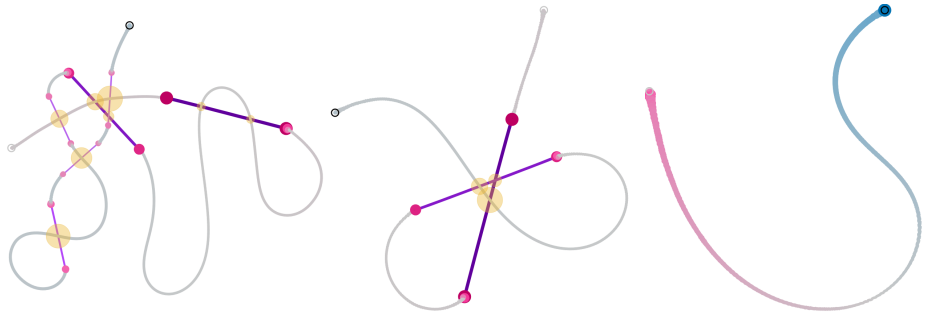


Figure 5.15 — Our approach applied to different t-SNE projections created with different perplexity values p for the same input data. It is noticeable that the number of jumps decreases. The same input data as for Figure 5.14 (d), a 3D spiral with varying radius, is used.

distances to the local neighborhood of the original space are too large in 2D. We could also confirm our distance-related observations in the different views in our interactive system. Additionally, when we think of the 3D spirals used as input, the distance between neighboring samples is always equal (Figure 5.14 (c)) or continuously increasing (Figure 5.14 (d)). Therefore, a jump is not present in the original data. For t-SNE, we observed that the number of such jumps decreases with a higher perplexity value; Figure 5.15 shows an example of different values for the input data of a spiral with increasing radius. With our approach, the jumps are clearly marked as artifacts; both the lines and adjacent samples are highlighted. The highlights at intersections of connections that are the result of such jumps show that most distances are rather high.

There is a difference in the distances and intersections visible in Figure 5.14 (b). In this example, the time series consists of multiple clusters. Here, jumps and intersections are also visible. However, most of the connections are very bright (or even green), indicating small distance errors (or a too short distance). Also, there is no or just a small highlight at the intersections, i.e., the estimated positions in multidimensional space are nearby. From our artificially generated cluster data, we also know that samples of the cluster are nearby, but there is a larger distance between the clusters.

Next, Figure 5.14 (d) shows an example of PCA with multiple intersections. Our approach marks these positions with a large distance since these intersections are not present in the original data (C4). In the underlying data (a spiral), no intersections are present. Without the highlight, one would have assumed that there might have been a temporal recurrence, especially as other projected samples are nearby. Many other examples where no intersection is present in the

original data show jumps in combination with intersections (e.g., Figure 5.14 (d) for t-SNE). When users are aware of this common pattern, they can examine such cases with care. Since in this case no jump is present, a misinterpretation is likely.

It might even happen that intersections are present in the original data but not projected. Figure 5.14 (a) shows the t-SNE projection of two intersecting lines. The projection created two sequences that turn by 90 degrees at the position where an intersection should be. The pink highlighted samples in the center indicate that there is an error in the projection. Since the sequences of samples do not go in a straight direction, the local neighborhood of samples changes in this area.

Figure 5.14 (e) shows examples of a change in speed (C1) on a logarithmic scale. While this variance is visible for PCA and MDS by the location of samples, it is not available for the other projection methods. Since t-SNE and UMAP do not preserve global distances, this is also expected. However, it may lead to misinterpretation. When exploring the temporal histograms, this behavior can be detected.

Finally, a sample projected to the wrong location (C3) is visible in Figure 5.9 B for t-SNE. This is a special case of a jump where only one sample is projected farther away. The underlying data represents a spiral without jumps or outliers.

Simulation Data: Kármán Vortex Street

We analyze a simulation of the Kármán vortex street [257] created by an obstacle. The visualization of computational fluid dynamics data is a typical use case for visualization in computational sciences, and there is previous work that applied multidimensional time series projection to ensemble data [230].

The data consists of 801 time steps with 121,604 dimensions. The simulation was performed on a uniform grid with a size 301×101. Figure 5.16 shows some visual representations of the data. The dimensions are built from values for the x and y positions and a direction vector for each grid cell. We used the projection methods PCA, MDS, t-SNE, and UMAP and the same quality metrics as in the previous section: q_{s1} (local neighborhood distances) for samples and q_{c2} (sliding window approach) for connections. Figure 5.17 shows some results for the projection, Figure 5.18 for the quality metrics and distances, and Figure 5.16 (left) an alignment between some visualizations of the data to one of the projection results.

From the analysis of the distance changes in the histogram for the original data (Figure 5.18 (Original Distance)), we notice three distinguishable regions: First, there is a short interval where samples are close together (distances between

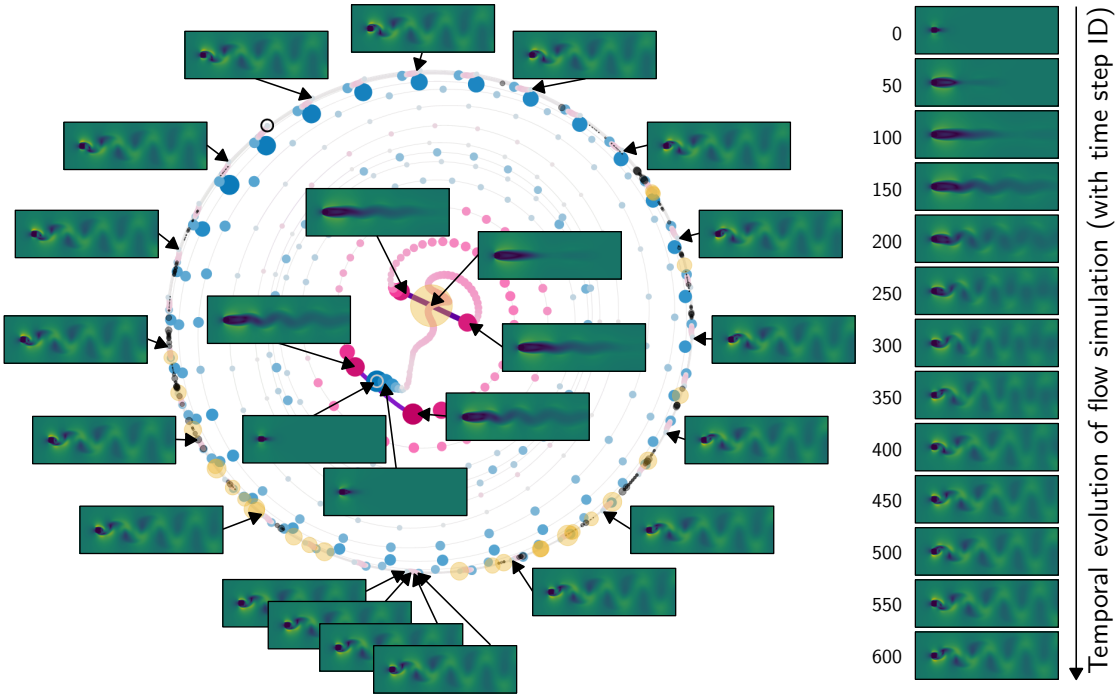


Figure 5.16 — Left: MDS projection of the Kármán vortex street simulation (same visualization used as in Figure 5.17). Images of the underlying data for samples are added to show the oscillating behavior and areas with jumps and intersections. Right: Visualization of some current states for the Kármán vortex street simulation representing the values for grid cells. Images are shown until sample 600. Already from sample 350 (until sample 801), no significant change to the cyclic behavior is present.

neighboring samples are rather small), then there is a transition phase where the differences increase. Finally, the distances are rather large in the last part and stay about the same for the second half of the timeline. These parts are present and distinguishable in the underlying data and represent the different stages, also visible in Figure 5.16 (right): (1) start phase without oscillation, (2) start of oscillation, (3) repeated pattern of regular oscillation (every 31 samples a new cycle starts). These three parts can roughly be separated into the ranges of the samples until 120 (first part), until 350 (second part), and until the last sample (third part); the transition to the next phase is gradual, therefore there is no exact moment of transition.

With MDS (Figure 5.17 A), the visualization clearly shows pink colors in the center, blue ones in the middle, and rather gray ones on the outside of the projection. It can be seen that distances in the center are too large, then rather too small, and finally well preserved. This is also visible in the temporal

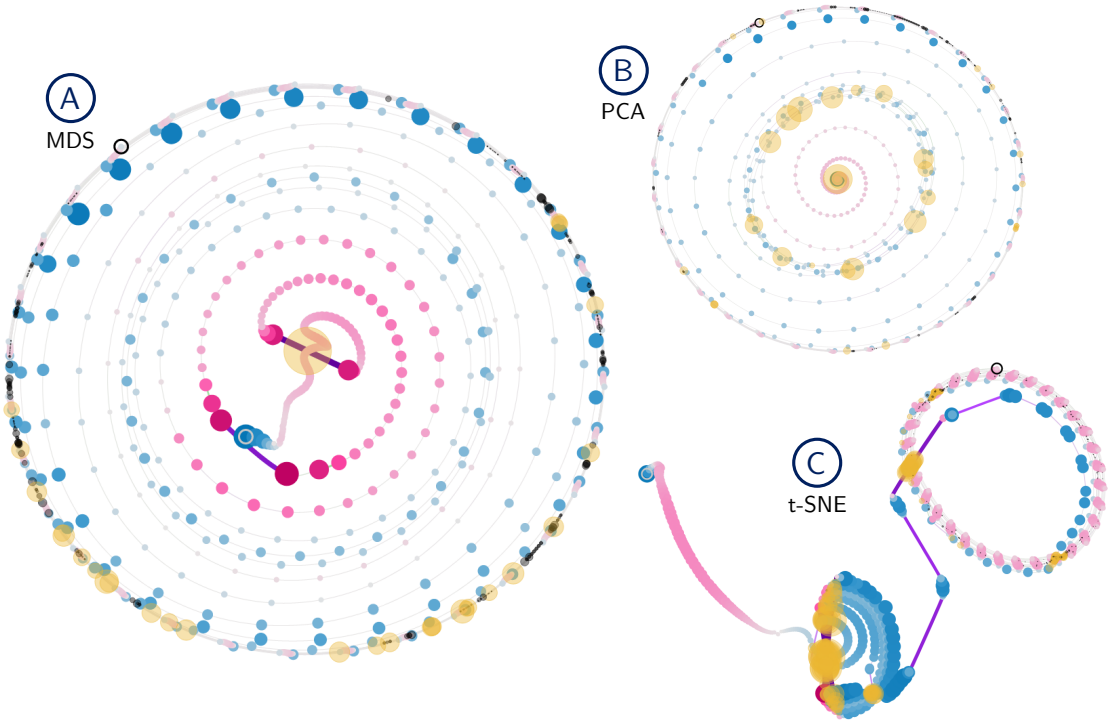


Figure 5.17 — Projections for the Kármán vortex street simulation: MDS (A) on the left, PCA (B) on the top right, and t-SNE (C) bottom right.

evolution shown in Figure 5.18 (2D Distance). Additionally, there are a few blue samples where the distances are too small at the very beginning. There are two jumps highlighted in purple since the distance is too large in 2D. These jumps are also visible in Figure 5.18 (Original Distance, 2D Distance, Shepard Diagram, and Distance Metric for Connections). For the first jump, both the connecting line and the adjacent samples are marked, and there is also an intersection visible. The distance at this intersection is rather large, and by exploring the histograms, we can verify this intersection as a projection error. In the histogram for the distance at intersections (Figure 5.18 (Original Distance at Intersections)), we see that the distance is more than two times the maximum distance between all subsequent samples. In Figure 5.18 (Original Distances), it is also visible that for distances in the original space, the high distances are at the end of the timeline, and therefore, this intersection (located at the beginning where subsequent samples are closer together in the original space) is very large compared to the local area. There is another jump without intersections slightly afterward. In the center of Figure 5.16 (left), it is also visible that highlighting these samples and connections is justified. While the samples connected by a jump are similar, samples crossing a jump or near a jump are pretty different.

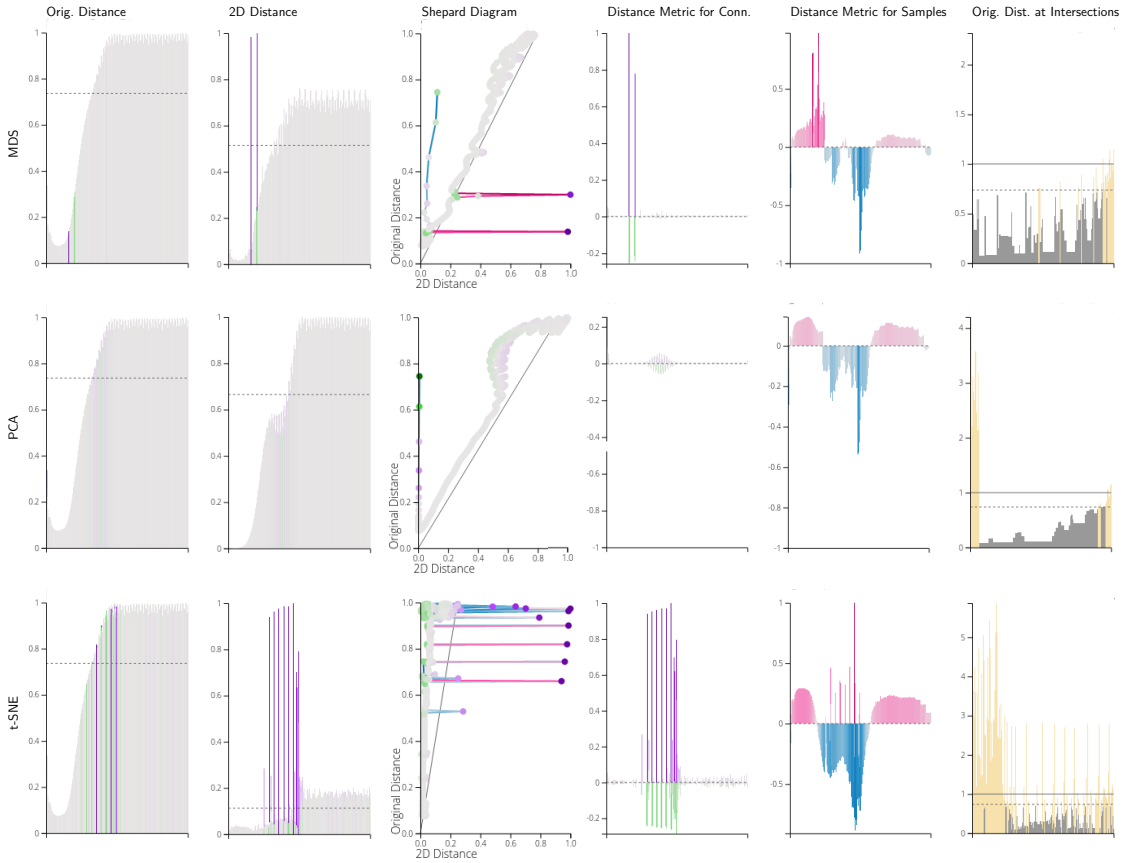


Figure 5.18 — Different plots for the projections presented in Figure 5.17: distances between neighboring samples in the original space and 2D, Shepard diagram for subsequent samples, quality metrics used for the samples (q_{s1} – local neighborhood distances), their connections (q_{c2} – sliding window approach), and temporal overview of distances at intersections.

Except for these errors, there are other intersections on the outside (many gray disks and a few yellow ones), but compared to the first ones rather small. These intersections are the result of many oscillating dots on the outside with rather small changes. In Figure 5.16 (left), the cyclic behavior of the underlying data is visualized by showing some of the 31 visualizations that belong to one oscillation. Additionally, at the bottom, the visualizations for a nearby position of different cycles show very similar representations.

For PCA (Figure 5.17 B), the quality was quite good, with rather low projection errors along the timeline. The beginning and end of the projection are well preserved when comparing the distances, except for the very beginning; here, the projected distances are too small. This is the case for both MDS and PCA: a few samples are blue and well visible as outliers in the Shepard

diagram (Figure 5.18). There are many intersections with a rather large error in the middle of the timeline and visible in the temporal plot for intersections (here, visible rather at the beginning since there were not many intersections before). Additionally, there are intersections in the last part of the projection, but they are again rather small (gray dots).

The t-SNE projection is shown in Figure 5.17 C. Here, the three parts of the simulation are clearly visible since they are projected to separate regions. The beginning of the simulation (the left part) gets more space. Due to the non-linearity, the size of the part in the center is larger, but the area at the end is comparable to the beginning. Intersections occur, especially in the middle part. Since the behavior should be rather cyclic, there is an inflow and an outflow to this region (resulting from the clustering behavior of t-SNE), and it was impossible to project this part without intersections. Therefore, many jumps are visible that intersect with the first part of the projection. In the transition to the last part of the simulation, multiple jumps are visible until, again, a cyclic behavior is visible.

We also created a projection with UMAP. However, the visualization was cluttered by intersections. While the first part was well projected, there were many intersections throughout the whole remaining timeline.

Real Footage: Hurricane Dorian Timelapse

As an example of real data, we explore satellite footage of hurricane Dorian[1] from the National Oceanic and Atmospheric Administration (NOAA). We used 95 frames (see Figure 5.19 (top) for some examples) with a resized resolution of 512×361, resulting in 554,496 dimensions created from each pixel's RGB values. Figure 5.20 shows PCA and MDS projections. For samples, metric q_{s2} (local neighborhood preservation) is used, and for connections q_{c1} (sequential distances). The more saturated and larger pink samples are, the more their local neighborhood differs in 2D and in the original space. Purple connections are too large compared to all other distances and green ones are to small. To compare to the underlying data, Figure 5.19 (bottom) shows some video frames used for the projection. Here, the beginning and end, different day and night scenes, and some transitions are added as annotations. In the projections, there is a flow from bottom to top visible in all images, reflecting the hurricane's movement. Additionally, there are many jumps available between the left and right sides. In contrast to some other examples we showed, most of these jumps are actually reflecting a strong change. We can see that one side of the projection represents daylight footage, whereas the other one night images (also, see Figure 5.19). In

[1] https://www.youtube.com/watch?v=e3g7NpCkZMM

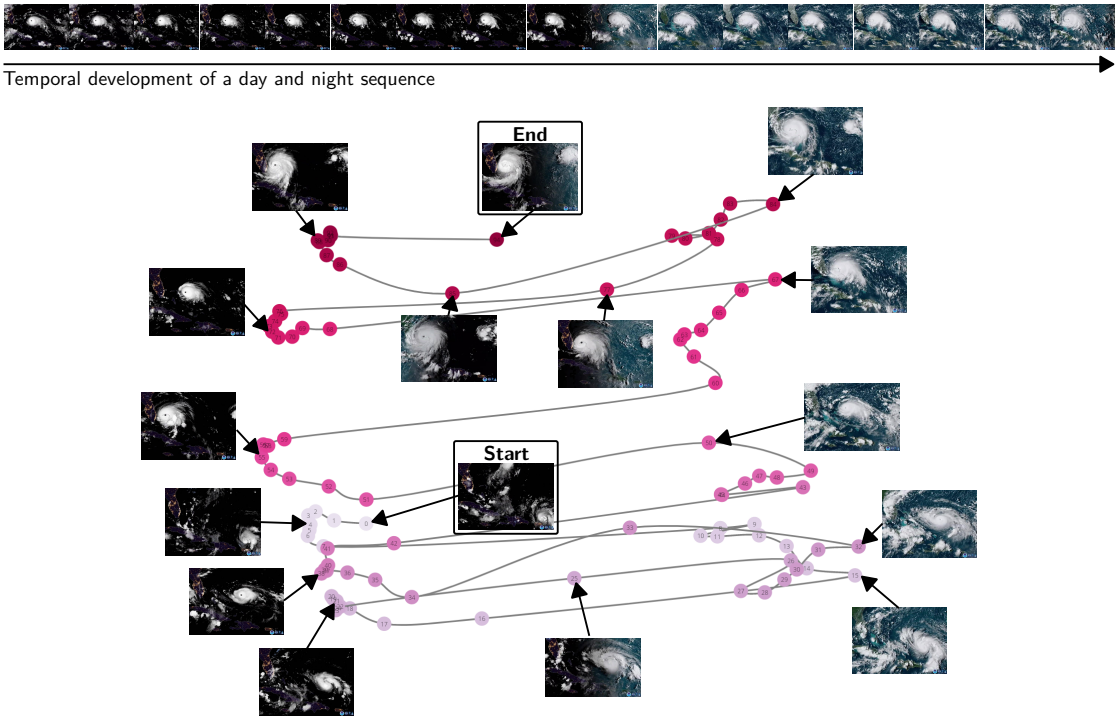


Figure 5.19 — Top: temporal evolution of one day and night sequence (the fourth one) of hurricane Dorian. Bottom: MDS projection for the hurricane Dorian data and some images used for the respective data samples. The color of the samples shows the temporal evolution. All images of hurricane Dorian are extracted from the footage of hurricane Dorian from the NOAA (https://www.youtube.com/watch?v=e3g7NpCkZMM). One frame of the video was extracted every second. The NOAA Satellite & Information Service owns the copyright of the footage.

total, we counted five daylight scenes, six night scenes, and eleven transitions. This structure is well visible in the projections and the footage. For the left PCA representation in Figure 5.20, the distance differences are similar for the transitions. Changing the quality metric for the connections such that it is assumed that the mean value is the best value the result in the middle column is created. Here, it seems that there are many jumps. However, a comparison in the other views shows that there is simply a very large difference between all smaller 2D distances and the larger ones. We see that it is important to verify highlighted areas in the main visualization with the other visualizations and that it is important that an appropriate quality metric and scaling thereof (especially, what is assumed to be the best value) is chosen.

For MDS (Figure 5.20 bottom), the distance differences between the outer and middle parts are better preserved (almost every distance difference is slightly

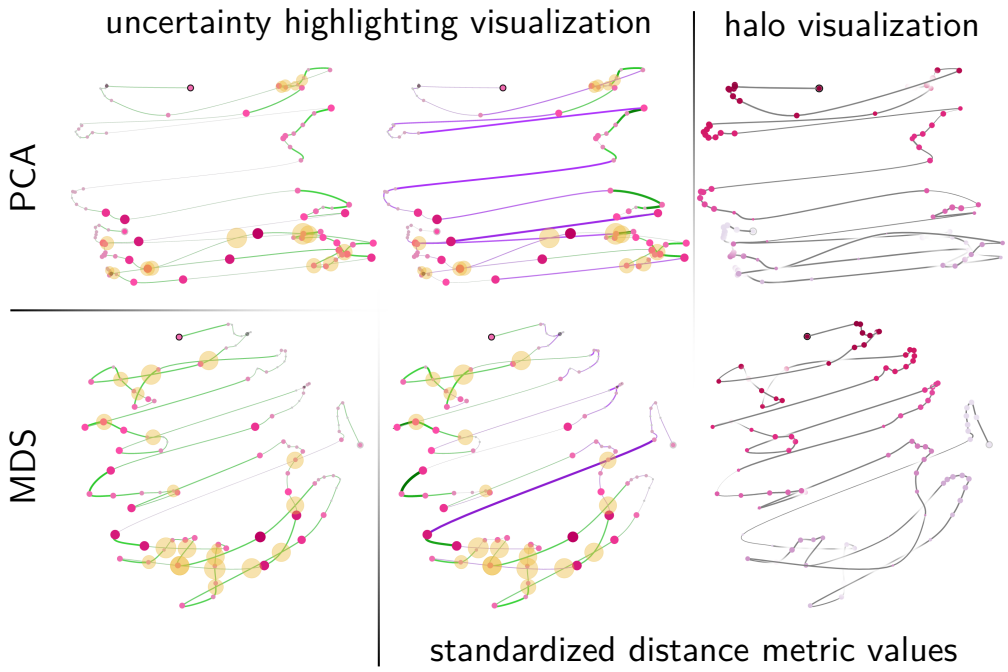


Figure 5.20 — PCA (top) and MDS (bottom) projections for the hurricane Dorian data. For samples, the quality metric reflecting the preservation of the neighborhood is used. For connections, the distance difference metric is used. In the middle and right column, metric values for connections are standardized (mean metric value is assumed as the best one), and in the left one, initial metric values are used. On the right, halo visualizations are shown.

too small in the left column). Using standardization to change the mean distance difference to the best value (Figure 5.20 bottom middle), one purple curve is shown. This distance is too large compared to all the other ones. Samples are drawn according to the local neighborhood preservation, and it is apparent that this is better preserved at the end (top) than at the beginning (bottom) (the same was also observed for PCA).

For both projection methods, in the area with more missing neighbors (at the bottom), there are also more intersections available. The differences at these intersections are near the mean value of all distances. The reason for these intersections is that more samples are projected in the bottom part of the projection since they are more similar than the samples in the later part of the time series. Figure 5.19 shows that the first three day and night sequences are similar, while the last one differs quite. This is also reflected in the projection. Since more similar samples (the first three sequences) are projected nearby, preserving distances and the neighborhood and avoiding intersections correctly is more challenging. Additionally, the first and third sequence are slightly

more similar than to second one, creating additional line crossings in the projection.

The images on the right in Figure 5.20 show the halo visualizations. The more uncertainty there is for samples, the smaller corresponding dots are drawn, and regions with intersections and, therefore, larger uncertainty are faded. Comparing the different visualization approaches, we can recognize that the first both columns are better suited to explore problematic regions of the projections. In contrast, the halo visualization uses less additional information (colors for lines and connections, highlighted intersections) to focus on well-projected points.

5.2.5 Discussion

In the following, we describe some observations we made while exploring artifacts with our approach.

Intersections and Jumps

With our interactive analysis system, we are able to successfully differentiate and verify if intersections and jumps available in the projection exist in the original space. In the case of an intersection or a close position, no highlight was drawn, or it was small and gray. It was yellow if the distance was rather large. Even then, a sample could have been in the neighborhood; this could be evaluated by comparing the distance distribution in the histograms. We could verify our observations with reference to the original data (e.g., 3D data or images).

Interactive Visualization

While it is possible to use different quality metrics for samples and connections, the visualization was best perceived if only one metric was applied. The samples and lines are usually rather small, and many samples may be visualized depending on the dataset. When the projection has too much overplotting, the visual encoding for samples and lines in the projection is not very helpful. However, the exploration of histograms is useful in this case. The temporal evolution of metrics in histograms supports the detection of outliers, sudden or continuous changes along the timeline. While the Shepard diagram shows the development of the relationship between 2D and the original distances, the linkage to the histograms shows more clearly where, for example, outliers are temporally located.

Depending on sample connections—straight lines or curves—slightly different intersections and intersection positions are visible that might fit better or worse. The use of highlights shows the distances of approximated positions in the multidimensional space. These distances are often much larger than one would assume from the projection. The distance differences we noticed for different connection types also indicate that there might be an even better path for the connections and, therefore, locations for the placement of intersections, i.e., the position where the distance is lowest or the actual position of an intersection. Determining this position is not trivial due to the nonlinear properties of some projection methods and the unknown change in multidimensional space between time steps.

Projection Method

We noticed that projection results and artifacts strongly depend on the datasets, projection method, and parameters. For the same dataset, we obtained large variability in results with different projection methods or parameter settings. This includes, for example, the occurrence of jumps that artificially divide the temporal data into multiple clusters, while the real data shows a relatively continuous development. This could be interpreted as strong changes along the time while it may actually be an artifact from the projection.

Additionally, we noticed random samples were occasionally projected to the wrong locations (Figure 5.9). Such samples appear as outliers within the temporal flow. They could also be interpreted as a sudden strong temporal change. Also, samples were sometimes projected around a virtual line (e.g., MDS projection in Figure 5.14 (e)). A more accurate projection would have placed them on the line. Such an observation could wrongly be interpreted as oscillation. With our visualizations, we were able to detect and explore intersections, jumps, and outliers.

Halo Visualization

The visualization of white faded lines at intersecting lines works well for data with a rather small number of samples (see Figure 5.9 or Figure 5.12). Once there are crowded regions with many intersections and samples close together, as, for example, in our example for simulation data (page 187 in Section 5.2.4), the halos may hide too much information, and the visualization might look blurred and broken. It is possible to reduce the size of halos; however, then they might be too small to be perceived.

5.2.6 Conclusion and Future Work

We characterized scenarios that may lead to misinterpretation of projected time series data. In particular, we argue that projection errors already present in traditional dimensionality reduction can become even more problematic for projection curves due to additional, prominent visual patterns. We address this uncertainty from the visualization by introducing visualization techniques that support the visual analysis of projection artifacts along the temporal path, especially at intersections and jumps in 2D space. We consider different pairwise and point-oriented quality metrics to identify areas in the visualization that might lead to misinterpretation. Furthermore, we estimate distances in the original data for intersection locations in 2D, allowing us to differentiate between intersections present only in the visualization and those also present in the data. Our approach can be applied to any projection method and might facilitate revealing their drawbacks for the projection of specific temporal datasets at hand. Especially researchers using dimensionality reduction methods on their temporal data can use our approach to verify their findings.

In the future, we would benefit from adapted or enhanced projection methods that would avoid temporal artifacts leading to misinterpretation by putting more weight on the temporal component, i.e., the temporal order of samples. These methods should generate especially low projection errors along the temporal path in the projection. Eventually, our system could be used to explore which dimensionality reduction method suits best for temporal data. A systematic evaluation could be performed to discover problems of projection methods and whether they depend on the data type.

However, our main message is: we would like to increase awareness for the additional visual uncertainty that comes with projecting multidimensional time series to 2D visualizations, which is an often used technique in different domains.

5.3 Summary and Conclusion

This chapter presented two visual analysis approaches in the context of projection methods. While the first one uses dimensionality reduction to explore multidimensional sequential data from ML by projecting it to 2D, the second one supports analyzing the usefulness of such a visualization method for temporal data. The first approach explored internal information of a classification model based on LSTMs for text sequence classification tasks. Different visualizations are provided to analyze the prediction process and better understand wrong classifications interactively. A focus lies on the visual representation of prediction results and hidden states during the prediction process. The second approach presents two visualization approaches to avoid misinterpretation of visualizations created for projections of multidimensional temporal data: The first visualization method highlights problematic regions of the projection and visualization; the second one fades such regions such that users can better focus on well-projected data. Additional diagrams for distances and metrics linked to the main visualization help better understand the visualized information in the main view.

While all research questions are handled in this chapter, a particular focus lies on RQ1, RQ3, and RQ4: the visual analysis of sequential data, the interpretability of ML, and the use of dimensionality reduction for the creation of visualizations. For RQ1, both approaches showed that the interactive exploration of the sequential data using multiple views can be helpful. Users are not restricted to working with a static image. Instead, they can explore and compare the results using other visualization forms, include numerical values (e.g., through tooltips), and change the parameter settings for the visual representations. We noticed that often, one visualization alone was not sufficient and the linkage between the different visualizations helped in the exploration. Similar to the approaches presented in the previous chapter (Chapter 4), the first approach of this chapter contributed to RQ3. Our approach helped by exploring the internal states of the ML model to better understand prediction behaviors, especially why some predictions were wrong or uncertain during the prediction process. The second approach was primarily designed to explore RQ4. Already in the first approach, projection results created with dimensionality reduction were used to interpret the data, but no direct statement about the usefulness regarding reliability was made. Nevertheless, we were able to successfully explore the temporal developments within the hidden state space. Additionally, appealing images could be created to show the shape of the projected hidden state space. Only the second approach explored the usefulness of such visualizations when analyzing sequential data. Projection errors and the visualization itself can lead

to misinterpretation of the underlying data. Therefore, interpreting such data must be done carefully, and the visualization approaches presented here can help avoid misinterpretations. However, the presented visualizations cannot show every problem of the projection in a satisfactory way. The effectiveness of the visual representations highly depends on the number of projected data samples, their location in 2D space, the projection method, and underlying structures.

CHAPTER 6

Conclusion

In this thesis, different approaches for the visual analysis of sequential data were presented. The focus was on eye tracking data and analyzing internal states of ML models. Additionally, the use of dimensionality reduction for the generation of visualizations for sequential data played an important role.

The following presents a summary of the chapters, a discussion of the research questions posed earlier (Section 1.1), and future research directions.

6.1 Summary of Chapters

Except for the first two chapters, containing an introduction and background information, this thesis is structured into three main chapters introducing different visual analysis approaches for sequential data. Chapter 3 explores temporal data from eye tracking in two different systems. The subsequent chapter (Chapter 4) introduces two approaches designed to foster the explainability of ML models using sequences as input, in internal mechanisms, and, in the first approach, as output. In the last chapter (Chapter 5), dimensionality reduction techniques are applied to sequential data (in one case from ML) as preparation for visualization.

Section 3.1 – Visual Exploration of Microsaccades The main idea behind the first system presented in this thesis, VisME, is to provide an interactive visualization system where users can use a filter to detect (micro)saccades with

different filter settings to analyze the respective results in multiple views. This allows users to try different parameter settings for eye movement detection and explore the influence on the visualizations. Additionally, other researchers can use the tool on some dataset to verify that respective parameter settings are appropriate. Consequently, the tool supports reproducing results in the eye tracking community.

Section 3.2 – Comparative Gaze Analysis ETFuse is a system to compare the eye movements of two people playing an online game against each other or in collaboration. The users play on two devices with possibly different hardware properties, and the devices may be located in different places. This approach supports a flexible setup of the experiment using available hardware while making the data comparison simple by automatically merging the data for comparison. Here, temporal and spatial synchronization is performed, and different sampling frequencies of the recordings are considered. Visualizations include gaze plots, attention maps, and a timeline visualization to compare the distance of gaze positions of the two players.

Section 4.1 – Neural Machine Translation There are still many issues when automatically translating a document that may result in poor translation results. In the visual analytics system NMTVis, the capabilities of humans and ML are combined to create high-quality translations. Users can interactively explore documents and sentences, verify their correctness, understand the translation process, correct the sentences, and fine-tune the translation model. The visualizations to show sequential internal information of the underlying ML model include attention views to visualize the relationship between words of the source sentence and the current translation. Additionally, the beam search view visualizes different translation possibilities as a hierarchical structure and provides insights into how the underlying translation model chooses the final translation.

Section 4.2 – Visual Question Answering The VQA Explorer uses scene graphs representing the content of images and questions as input to predict answers. The scene graphs consist of nodes and edges that represent objects in an image and their relations to each other. Each object can have attributes such as the color or its current state. The edges have properties describing the connections between objects. Our approach aims to identify and correct prediction errors and provide insights into the prediction process through visualization of internal information. The approach can be used for debugging and data curation.

Section 5.1 – Analysis of Hidden States In an interactive approach for exploring the internal states of a text classification model, users can explore how hidden states store and process information while feeding an input sequence into the ML model. The main view shows the projection generated with dimensionality reduction for the hidden states in the LSTM layer. The trajectory in the projection can show how the hidden states change when the model processes the words in a text sequence. The sequential evolution can also be shown in other visualizations highlighting the expected prediction result or distances between hidden states.

Section 5.2 – Visual Quality of Time Series Projections Finally, an approach for reducing misinterpretation of visualizations for 2D projections is presented. The projections are generated with dimensionality reduction techniques applied to multidimensional temporal data. When data is being projected to 2D, not all information can be maintained. Well-known problems are false and missing neighbors and variances in distances between neighbors. For temporal data, additional connections between samples exist in the visualizations. They may introduce additional artifacts and possibilities for misinterpretation. For example, varying distances, jumps, outliers, and intersections could be present in the projection but not in the data. Our tool has two visualization modes with different goals for exploring such artifacts. An uncertainty highlighting visualization highlights problematic regions of the projection and the halo visualization directs focus to the correctly projected data.

6.2 Overarching Discussion

For all presented projects, different visual analysis systems were developed that use sequential data as input. The approaches have the goal of allowing interactive exploration of the visualizations for the input data. Most of these analysis systems were designed for two main domains: eye tracking and explainability of ML with sequential aspects. For eye tracking, the approaches support parameter exploration as well as reproduction of eye movement detection. Additionally, they facilitate comparative analysis of eye movement data. In the context of ML, models were analyzed using interactive visualizations and, e.g., unsupervised learning to explore the internal prediction mechanisms and improve the performance for predictions (e.g., through improved models or input data). Additionally, challenges in the visualization and analysis of time series data were explored, namely for multidimensional time series projections generated through dimensionality reduction. All our techniques profit in numerous ways from interactive visualizations. Besides classical visualizations such as bar

charts, timeline visualizations, and scatterplots, our visualization techniques include matrix representations, heatmaps, and scanpath visualizations. We also use tree- and graph-based visualizations and parallel coordinate plots. All visualizations were adapted for the specific use cases and a combination of different visualizations were incorporated into systems for visual analysis with different interaction techniques. We made the source code and additional material (e.g., datasets and videos) publicly available for all our approaches to allow reproducibility of our experiments, promote trust in our techniques, and foster open science.

In the following, we comment on the research questions posed at the beginning of this thesis (Section 1.1). The first research questions are more general and, thus, covered by more chapters of this thesis than the latter, more specific research questions.

RQ1: *How beneficial is interactive visual analysis in exploring sequential data?*

The usefulness of visual analysis for sequential data depends on the data, analysis goals, target groups, chosen visualizations, interaction techniques, and other factors. If no visualizations are available, users often face a large amount of raw data that requires analysis, which might be tedious. Even when static visualizations are provided that show important aspects of the data, users cannot explore the data from different perspectives. However, with interactive visual analysis approaches in general, data can be explored and adapted, allowing for better interpretation and conclusions to be drawn from the data. Techniques such as brushing and linking [332] combined with multiple coordinated views [273] can help to improve understanding visual elements when exploring them in a different context or from various perspectives with additional information. It is, for example, possible to provide more context in some visualizations. Additionally, parameters can be changed (see RQ2) for preprocessing and to obtain adapted visualizations (e.g., to modify the appearance of visual elements or show subsets of the data).

This research question concerns almost all presented methods of this thesis; in these approaches, sequential data plays an important role during the visual analysis. The usefulness of individual visual analysis approaches can be assessed through evaluations. Evaluation methods (see Section 2.7 for an overview) can include the demonstration of use cases, computer-based evaluations to determine the relevance of visualized information, and user studies (e.g., with experts) to obtain feedback on the actual use of an approach.

For each approach introduced in this thesis, we provided use cases to show the capabilities of our approaches and how our systems can be used. Here, the visual analysis was always helpful in exploring the data or improving the presented information. We noticed that the combination of different visualizations, interaction techniques, and tooltips helped us better understand the main visualization. They provided additional information (e.g., additional numerical data) and showed the data in different contexts (e.g., filtered data, with temporal or spatial focus). In some approaches, it was possible to even improve the quality of the provided data.

Additionally, computer-based evaluations revealed the usefulness of the presented information and mechanisms of the two approaches presented in Chapter 4. For example, we could show that iteratively improving translations and fine-tuning the ML model could create better predictions. This way, we could confirm that our proposed workflow succeeds in correcting sentences and generating a better translation result after improving the model with corrected sentences.

In user studies, usually performed with experts of related domains, a user-oriented analysis can be performed to see if the visual analysis is helpful in practical use. The experts provide feedback on how useful a visual analysis system is in exploring (sequential) data. We performed user studies for the approaches presented in Chapter 3 and Chapter 4. Each user study confirmed the usefulness of our approaches. However, creating a visual analysis system that supports all user needs is challenging. Experts and beginners alike require tailored tools to effectively interpret and interact with the data. Domain experts often need extended features such as advanced filtering and interaction techniques to adapt the visualizations or the presented data to align with their specific needs. Such tools help experts to focus on complex behaviors while beginners profit from a simpler and more intuitive system design. In the user studies performed during this thesis, we always received positive feedback for our prototypes, but there were also always recommendations for improvement. For example, VisME (Section 3.1) was evaluated with a think-aloud protocol, and overall, all participants liked our approach and highlighted some of our interaction techniques. However, they also mentioned that additional features would be nice for the analysis (e.g., manual modification of samples belonging to microsaccades). While our approach is a good starting point for the analysis, some of the users would prefer to use scripts or other tools for a deeper analysis. This highlights that additional features would be required for the whole visual analysis process. In ETFuse (Section 3.2), an expert analyzed the collected data. He noted that not all visualizations were immediately helpful but could provide

additional information. For NMTVis (Section 4.1), participants mentioned that more automatic help finding incorrect translations would be useful.

Overall, the visual analysis of sequential data can be helpful, but the specific needs and requirements of users might require individual features for a successful visual analysis of their data. Depending on the experience levels of users and analysis goals, every user might have individual needs for the analysis. This was visible in the user feedback we collected in our experiments.

> **RQ2**: *How does the choice of (hyper)parameter values (in preprocessing and the analysis) affect the visual analysis of sequential data?*

In the approaches presented in this thesis, parameter settings can be modified for data preprocessing, analysis, and visualization. Well-chosen values can help users better understand the data, but poor values can lead to misinterpretation and wrong conclusions. It is often difficult to recognize and verify that values are well chosen; especially when there is no comparison possible, when chosen values are not provided in published work, or when others cannot reproduce the experiments. The parameter settings can influence the data used for visualization and how it is presented to users and afterward interpreted. Therefore, this research question has a direct connection to the previous one. Most datasets have to be preprocessed before visualization. This can be achieved in multiple ways and with different options. Additionally, for the visualizations, various decisions have to be made. Sometimes, these options are passed on to users who can adapt different parameters. This includes, for example, the size and amount of visualized data elements or other thresholds.

This research question concerns all presented methods. For some, it plays a major role (e.g., Section 3.1), and for others just a minor one. For example, VisME (Section 3.1) has a special focus on the preprocessing step before visualization. Users can interactively set the parameters for the (micro)saccade filter. We noticed that different settings influence both the amount and length of the detected eye movements and, subsequently, the visualizations and conclusions that can be drawn from them. Therefore, it is important that the parameters are selected with care and that the visualized results are verified such that the detected eye movements are labeled correctly. Additionally, for the visualizations, e.g., the binning size for the rose plots plays an important role during interpretation. This affects how much data is averaged and whether, for example, outliers can be detected.

For ETFuse (Section 3.2), in the first step, the data from the different sources has to be merged. The user has to select a method and additional parameters

(e.g., the stone used for synchronization). This selection sometimes results in a wrong synchronization (e.g., when using poor-quality recordings or when visual elements on the screen prevent a successful synchronization). If the user does not notice this, wrong conclusions will be drawn from the information presented. Additionally, interval sizes must be chosen for the distance plots and attention maps to aggregate eye movement data. The smaller these intervals, the more information is revealed, but there is not much averaging, which could increase interpretability. If the intervals are larger, some important information may be missed.

In NMTVis (Section 4.1), the user has to first select the translation model. Of these, the Transformer creates better translations, and it is therefore more challenging to find mistranslations. Additionally, using this model, the connections between the underlying model and the prediction in the attention view are less visible. Our LSTM architecture usually creates more mistranslations, which makes it easier to find wrong translations and improve the model. The alignment in the attention view also makes more sense for the translations generated with the LSTM model compared to the ones created with the Transformer. Besides the translation model, visual parameters influence the presented data. For example, the beam size provides the capability to show more alternative translations, possibly including better ones. However, the screen size to show more translations may be limited, and users would have to process more information.

Similar to NMTVis, in the VQA Explorer (Section 4.2), different DL models can be selected. Additional visual parameters, such as size or color of visual elements, can be adapted to influence how the visual information is perceived and interpreted.

While for NMTVis and the VQA Explorer users have to select a DL model, the system to explore hidden states of classification results for text sequences (Section 5.1) requires selecting a projection method (along with its hyperparameters) during preprocessing. The projection method influences how the hidden state space is presented. In our examples, t-SNE showed the most suitable and best-looking results. Using, for instance, PCA would have projected all information into a narrower space, which would have made it impossible to follow the path of individual sequences. Additionally, this approach offers users different options for using the color of dots (e.g., actual vs. predicted classes) and lines (color vs. grayscale for better visibility), or the type of presented words on top of the projection.

In the system to explore the visual quality of projections of temporal multidimensional data (Section 5.2), the first choice is also the projection method that

influences the visualized elements. This choice affects how useful the presented visualizations are and how many projection errors, artifacts, or other components for misinterpretation are shown. Next, different metrics can be chosen for the visualizations to find problematic regions in the projection. Depending on the use case, one metric might work better than another to detect areas likely affected by misinterpretation in the projection. Additionally, there are settings for the size of visual elements that influence how the presented information is perceived.

Overall, well-chosen parameter settings are crucial for creating meaningful visualizations from input data and avoid misinterpretation. The visual representations can support users in interpreting the underlying data and in making decisions based on the provided information if the data is well presented.

RQ3: *How can visual analysis of internal (sequential) components of ML methods improve their interpretability?*

ML is currently becoming more relevant in our lives. Its methods can be very powerful but difficult to understand. There are already many approaches that help better understand ML models (see Section 2.5.3). However, there are still many challenges in interpreting the underlying prediction mechanisms and generating better predictions. The approaches presented in this work can help achieve goals such as generating a better understanding of how a prediction was made, debugging prediction results or mechanisms, data curation, and improving the predictions.

This research question concerns three methods presented in this thesis using ML as a subject for the exploration. NMTVis (Section 4.1) provides two types of visualizations for sequential internal information of the underlying models: In the form of attention-based visualizations, it is possible to see an alignment between two sentences in different languages. With the beam search view, it is possible to see how the prediction was performed by visualizing different alternative translations in a hierarchical structure. Additionally, parallel coordinate plots provide information about the prediction quality and can be used to find mistranslations.

The VQA Explorer (Section 4.2) does not provide internal information for sequences but for the scene graphs during prediction. Here, attention values for nodes and edges are presented. The attention values are reflected in the size of nodes and can show if the correct object has the highest attention. This helps in debugging or data curation by detecting missing or wrong objects, attributes, or connections in the scene graphs.

In the system to explore hidden states of classification results for text sequences (Section 5.1), information about hidden states and expected predictions can be shown to the user. This approach can be used to see how these values change while performing the prediction. Different visualizations are provided during the analysis. Additional information, such as words connected to large changes, can also be highlighted. The visualizations help to explore and understand wrong, but also correct or uncertain prediction results.

Overall, internal ML components can provide a better understanding of both correct and incorrect predictions. This can be achieved by providing user access to internal mechanisms and states of the ML models through interactive visualizations. The approaches can also help improve the data quality, prediction results, and prediction models. We noticed that when predictions were incorrect, it was not always the problem of a poorly trained model; often, it was the fault of poorly or faulty labeled data.

RQ4: *How appropriate are dimensionality reduction techniques for visually exploring sequential data?*

Various dimensionality reduction methods are used to project data to 2D for visualization. Many previous research projects also do this for sequential data to find typical sequential patterns, trends, and behaviors (see Table 5.1 on page 171). There is even research that describes how such properties can be identified in the visualizations of the projections [45, 318]. If a projection method is well chosen for some data to extract important information, the visualizations can provide meaningful insights into the structure of the data. However, in one of the projects presented in this thesis (Section 5.2), we observed that there are many issues with using such a visualization that could lead to wrong interpretations. In general, problems caused by dimensionality reduction are projection errors, including missing and wrong neighbors and varying distances. Additionally, lines connecting subsequent samples that might intersect are present in the visualizations for sequential data. At first sight, such visualizations seem to be intuitive. However, our investigation showed that they should be used with care. The apparent patterns mentioned in other research papers are not always existing in the data and can be misinterpreted.

This research question concerns only the approaches using dimensionality reduction methods on sequential data for the visual analysis; this includes the two techniques presented in Chapter 5. In the system to explore hidden states of classification results for text sequences (Section 5.1), we use the dimensionality reduction method to project the hidden state space to 2D. The visualizations

can then help explore how the prediction process was performed within this space. Here, it is crucial which projection method is chosen to create the visualizations. For us, it was important to provide a visualization that allowed us to follow the path of intermediate prediction steps. Our analysis did not explore whether the actual visualizations are reliable. Nevertheless, we noticed that the visualizations can help better understand how the hidden states change in the context of the hidden state space. The usefulness of such visualizations can be explored with our approach to explore the visual quality of projections of temporal multidimensional data (Section 5.2). We noticed that outliers, jumps, oscillating data samples, and otherwise varying distances visible in the 2D projection are not always existing in the data. Such artifacts often occur, especially for nonlinear projections. We also had cases with intersecting lines that did not represent temporal recurrences for linear projection methods (e.g., PCA). With our interactive visual analysis approach, misinterpretation can be circumvented by considering and examining problematic regions. Here, different distance and metric plots can be used in addition to a visualization that highlights or fades such artifacts. While through these visualizations misinterpretations can be avoided, the interpretation is still not easily done, and not all problems can always be detected. A better solution would be to use a specifically designed projection method for temporal data. Even then, due to the curse of dimensionality, there might be similar problems in the interpretation of the data.

6.3 Future Research Directions

In this thesis, different approaches related to sequential data analysis were presented. The main focus was on temporal data from eye tracking, the explainability of internal states of ML models based on sequences, and the use of dimensionality reduction to create visualizations for sequential data. There is still a lot of research left in each of these areas. Besides the future research directions mentioned for each approach, the following describes some broader research areas.

In the area of eye tracking, continuously, newer and better hardware is being produced. Newer technology might be able to record eye movements with higher frequency and accuracy, allowing the exploration of even smaller details. These explorations might require new analysis techniques compared to the methods used before. In our presented approaches, we already allow a flexible selection of parameter settings for the analysis (Section 3.1). Here, it would also be possible to evaluate if parameter settings for different recording frequencies need to be changed to correctly detect eye movements or if new methods are

required. Additionally, it is worth exploring whether the parameter settings depend on specific properties of the recorded data. Furthermore, ML approaches could be used to automatically determine the most suitable parameter values for specific datasets to ensure accurate eye movement detection. Finally, in our analysis approaches, we could integrate additionally recorded data (e.g., neurophysiological data) to investigate potential connections with specific eye movement behaviors, such as the occurrence of microsaccades.

For ML, the development of many new and powerful ML models could be observed during this research. Since ML is becoming increasingly popular, there is a large interest in integrating it into our lives. However, such a broad use requires trust, the reliability for creating correct prediction results, and to overcome the difficulties in understanding the prediction models. The interpretability of ML models is often still not well addressed, but it is highly needed to understand prediction mechanisms and build trust. The ML-based approaches presented in this thesis (Chapter 4 and Section 5.1) serve as small building blocks to achieve a better understanding of ML techniques. However, only some specific approaches are covered. Each approach also presents just a small portion of the internal mechanisms to users to draw conclusions. There are still many unexploited opportunities in this area. These could include the connection of the training data, data from the training process, internal mechanisms during prediction, and the final prediction. It is both important to understand why predictions are not correct and why some specific wrong prediction was generated. Here, a more detailed explanation of the decisions and results of the model is required, including data from various sources to support the reasoning. More research is required to solve the problems of XAI for general approaches and specific applications, especially for newer, larger, and more complex models. Individual ideas of existing approaches, also the ones presented in this thesis, can be extended or combined with new analysis approaches to understand these models and build trust. Such trustworthiness is particularly essential in fields like medicine, where it is crucial for physicians to understand why a specific diagnosis was predicted by a DL model for a patient.

Next, the visual analysis processes could be improved by integrating more automatic processing or even ML. For example, for NMTVis (Section 4.1), the search for mistranslations or the detection of poor scenes in the VQA Explorer (Section 4.2) could be performed automatically. Similarly, the parameter settings for microsaccades filtering or the detection of microsaccades themselves (Section 3.1) could be based on ML. Interesting eye movement patterns (such as described in the use case of our visual analysis system ETFuse (Section 3.2) could be detected automatically with ML. As introduced in Section 2.5.3 as

ML4Vis4ML, we already apply unsupervised ML on hidden states to generate visualizations for a supervised ML approach. The next step would be to extend this approach by applying some supervised ML approach on data generated with supervised ML. This idea might bring new and unexpected insights into different ML models. Since ML approaches can create astonishing results, new insights could be obtained. However, the disadvantage of such a method is that the trustworthiness of the results may again be unclear, as they would be generated by a new black box model.

When using dimensionality reduction methods, it is often unclear which method and which parameter choice are best suited to create the most appropriate visualizations required in the analysis. In the case of Section 5.1, t-SNE with some fixed parameter settings was chosen as preparation for the creation of the visualizations. Another method or parameter values might have generated better visualizations to analyze the prediction results of the ML models. A systematic evaluation could reveal which method and parameter settings is most suitable. Moreover, for the use of dimensionality reduction in the analysis of sequential data (see Section 5.2), an approach is needed that automatically generates a better projection for time series, considering temporal aspects. The visualization itself of projected data could then lead to less misinterpretation. Such an improved projection method for sequential data could be used to gain more reasonable insights into the properties of sequences. An extension thereof would be an adaptive method that automatically chooses the best-suitable dimensionality reduction method and sets parameter values based on the properties of the data to create visualizations with a minimum of distortions. The generated results could still be explored with the visualization approaches proposed here to evaluate if users can trust the presented information.

Author's Work

[1] F. Beck, M. Burch, T. Munz, L. Di Silvestro, and D. Weiskopf. Generalized Pythagoras trees for visualizing hierarchies. In *Proceedings of the 5th International Conference on Information Visualization Theory and Application (IVAPP)*, pages 17–28, 2014, doi: 10.5220/0004654500170028.

[2] F. Beck, M. Burch, T. Munz, L. Di Silvestro, and D. Weiskopf. Generalized Pythagoras trees: A fractal approach to hierarchy visualization. In S. Battiato, S. Coquillart, J. Pettré, R. S. Laramee, A. Kerren, and J. Braz, editors, *Computer Vision, Imaging and Computer Graphics - Theory and Applications*, pages 115–135. Springer International Publishing, 2015, doi: 10.1007/978-3-319-25117-2_8.

[3] M. Burch, T. Munz, F. Beck, and D. Weiskopf. Visualizing work processes in software engineering with developer rivers. In *Proceedings of the 3rd IEEE Working Conference on Software Visualization (VISSOFT '15)*, pages 116–124. IEEE, 2015, doi: 10.1109/VISSOFT.2015.7332421.

[4] M. Burch, T. Munz, and D. Weiskopf. Edge-stacked timelines for visualizing dynamic weighted digraphs. In J. Braz, A. Kerren, and L. Linsen, editors, *Proceedings of the 6th International Conference on Information Visualization Theory and Applications (IVAPP)*, pages 93–100. SciTePress, 2015, doi: 10.5220/0005259200930100.

[5] R. Garcia, T. Munz, and D. Weiskopf. Visual analytics tool for the interpretation of hidden states in recurrent neural networks. *Visual Computing for Industry, Biomedicine, and Art (VCIBA)*, 4(1):24, 2021, doi: 10.1186/s42492-021-00090-0.

[6] F. Heyen, T. Munz, M. Neumann, D. Ortega, N. T. Vu, D. Weiskopf, and M. Sedlmair. ClaVis: An interactive visual comparison system for classifiers. In *Proceedings of the 2020 International Conference on Advanced Visual Interfaces (AVI '20)*, article 9, pages 1–9. Association for Computing Machinery, 2020, doi: 10.1145/3399715.3399814.

[7] S. Künzel, T. Munz-Körner, P. Tilli, N. Schäfer, S. Vidyapu, N. T. Vu, and D. Weiskopf. Visual explainable artificial intelligence for graph-based visual question answering and scene graph curation. *Visual Computing for Industry, Biomedicine, and Art (VCIBA)*, 8(1):9, 2025, doi: 10.1186/s42492-025-00185-y.

[8] T. Munz. VisME software v1.2. Zenodo, 2019, doi: 10.5281/zenodo.3352236.

[9] T. Munz, M. Burch, T. v. Benthem, Y. Poels, F. Beck, and D. Weiskopf. Overlap-free drawing of generalized Pythagoras trees for hierarchy visualization. In *2019 IEEE Visualization Conference (VIS)*, pages 251–255, 2019, doi: 10.1109/VISUAL.2019.8933606.

[10] T. Munz, L. L. Chuang, S. Pannasch, and D. Weiskopf. VisME: Visual microsaccades explorer. *Journal of Eye Movement Research*, 12(6), 2019, doi: 10.16910/jemr.12.6.5.

[11] T. Munz, N. Schäfer, T. Blascheck, K. Kurzhals, E. Zhang, and D. Weiskopf. Supplemental material for comparative visual gaze analysis for virtual board games. DaRUS, V1, 2020, doi: 10.18419/darus-1130.

[12] T. Munz, N. Schäfer, T. Blascheck, K. Kurzhals, E. Zhang, and D. Weiskopf. Comparative visual gaze analysis for virtual board games. In *Proceedings of the 13th International Symposium on Visual Information Communication and Interaction (VINCI '20)*, article 11, pages 1–8. Association for Computing Machinery, 2020, doi: 10.1145/3430036.3430038.

[13] T. Munz, N. Schäfer, T. Blascheck, K. Kurzhals, E. Zhang, and D. Weiskopf. Demo of a visual gaze analysis system for virtual board games. In *ACM Symposium on Eye Tracking Research and Applications (ETRA '20 Adjunct)*, article 2, pages 1–3. Association for Computing Machinery, 2020, doi: 10.1145/3379157.3391985.

[14] T. Munz, R. Garcia, and D. Weiskopf. Visual analytics system for hidden states in recurrent neural networks. DaRUS, V1, 2021, doi: 10.18419/darus-2052.

[15] T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. Visual-interactive neural machine translation. In *Proceedings of Graphics Interface 2021 (GI 2021)*, pages 265–274. Canadian Information Processing Society, 2021, doi: 10.20380/GI2021.30.

[16] T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. NMTVis - Neural machine translation visualization system. DaRUS, V1, 2021, doi: 10.18419/darus-1849.

[17] T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. NMTVis - Trained models for our visual analytics system. DaRUS, V1, 2021, doi: 10.18419/darus-1850.

[18] T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. NMTVis - Extended neural machine translation visualization system. DaRUS, V1, 2022, doi: 10.18419/darus-2124.

[19] T. Munz, D. Väth, P. Kuznecov, N. T. Vu, and D. Weiskopf. Visualization-based improvement of neural machine translation. *Computers & Graphics*, 103:45–60, 2022, doi: 10.1016/j.cag.2021.12.003.

[20] T. Munz-Körner and D. Weiskopf. Visual analysis system to explore the visual quality of multidimensional time series projections. DaRUS, V1, 2024, doi: 10.18419/darus-3553.

[21] T. Munz-Körner and D. Weiskopf. Supplemental material for "Exploring visual quality of multidimensional time series projections". DaRUS, V1, 2024, doi: 10.18419/darus-396.

[22] T. Munz-Körner and D. Weiskopf. Exploring visual quality of multidimensional time series projections. *Visual Informatics*, 8(2):27–42, 2024, doi: 10.1016/j.visinf.2024.04.004.

[23] T. Munz-Körner, S. Künzel, and D. Weiskopf. Supplemental material for "Visual-explainable AI: The use case of language models". DaRUS, V1, 2023, doi: 10.18419/darus-3456.

[24] T. Munz-Körner, S. Künzel, and D. Weiskopf. Poster: Visual-explainable AI: The use case of language models. In *International Conference on Data-Integrated Simulation Science (SimTech2023)*, 2023, URL: https://www.simtech2023.uni-stuttgart.de/documents/Theme-2/Munz-Koerner-Tanja.pdf.

[25] N. Schäfer, S. Künzel, T. Munz-Körner, P. Tilli, S. Vidyapu, N. T. Vu, and D. Weiskopf. Visual analysis of scene-graph-based visual question answering. In *Proceedings of the 16th International Symposium on Visual Information Communication and Interaction (VINCI '23)*, article 25, pages 1–8. Association for Computing Machinery, 2023, doi: 10.1145/3615522.3615547.

[26] N. Schäfer, P. Tilli, T. Munz-Körner, S. Künzel, S. Vidyapu, N. T. Vu, and D. Weiskopf. Visual analysis system for scene-graph-based visual question answering. DaRUS, V1, 2023, doi: 10.18419/darus-3589.

[27] N. Schäfer, P. Tilli, T. Munz-Körner, S. Künzel, S. Vidyapu, N. T. Vu, and D. Weiskopf. Model parameters and evaluation data for our visual analysis system for scene-graph-based visual question answering. DaRUS, V1, 2023, doi: 10.18419/darus-3597.

[28] N. Schäfer, S. Künzel, P. Tilli, T. Munz-Körner, S. Vidyapu, N. T. Vu, and D. Weiskopf. Extended visual analysis system for scene-graph-based visual question answering. DaRUS, V1, 2025, doi: 10.18419/darus-3909.

Bibliography

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. [Online]. Available: https://www.tensorflow.org/.

[30] R. V. Abadi and E. Gowen. Characteristics of saccadic intrusions. *Vision Research*, 44(23):2675–2690, 2004.

[31] C. C. Aggarwal et al. *Neural Networks and Deep Learning*. Springer, 2018.

[32] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47–60, 2007.

[33] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Springer Nature, 2023.

[34] J. Albrecht, R. Hwa, and G. E. Marai. The Chinese Room: Visualization and interaction to understand and correct ambiguous machine translation. *Computer Graphics Forum*, 28(3):1047–1054, 2009.

[35] M. Ali, M. W. Jones, X. Xie, and M. Williams. TimeCluster: Dimension reduction applied to temporal data for visual analytics. *The Visual Computer*, 35(6–8):1013–1026, 2019.

[36] G. Alicioglu and B. Sun. A survey of visual analytics for explainable artificial intelligence methods. *Computers & Graphics*, 102:502–520, 2022.

[37] S. Almeida, Ó. Mealha, and A. Veloso. Interaction behavior of hardcore and inexperienced players: "Call of Duty: Modern Warfare" context. In *Proceedings of SBGames*, pages 106–115, 2010.

[38] S. Almeida, A. Veloso, L. Roque, and O. Mealha. The eyes and games: A survey of visual attention and eye tracking input in video games. In *Proceedings of SBGames*, pages 1–10, 2011.

[39] R. Andersson, L. Larsson, K. Holmqvist, M. Stridh, and M. Nyström. One algorithm to rule them all? An evaluation and discussion of ten eye movement event-detection algorithms. *Behavior Research Methods*, 49(2): 616–637, 2017.

[40] G. Andrienko, N. Andrienko, M. Burch, and D. Weiskopf. Visual analytics methodology for eye movement studies. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2889–2898, 2012.

[41] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

[42] A. Appel, F. J. Rohlf, and A. J. Stein. The haloed line effect for hidden line elimination. *SIGGRAPH Computer Graphics*, 13(2):151–157, 1979.

[43] M. Aupetit. Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 70(7):1304–1330, 2007.

[44] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A review of temporal data visualizations based on space-time cube operations. In *Eurographics Conference on Visualization*, 2014.

[45] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1): 559–568, 2016.

[46] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

[47] A. Bahill, M. R. Clark, and L. Stark. Glissades—eye movements generated by mismatched components of the saccadic motoneuronal control signal. *Mathematical Biosciences*, 26(3):303–318, 1975.

[48] S. A. Balasubramanian. Visual analysis of a machine learning approach applied to turbulence data, 2022, Master's thesis, University of Stuttgart, doi: 10.18419/opus-12790.

[49] M. B. Barbara Tversky, Julie Bauer Morrison. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[50] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[51] L. Battle, M. Angelini, C. Binnig, T. Catarci, P. Eichmann, J.-D. Fekete, G. Santucci, M. Sedlmair, and W. Willett. Evaluating visual data analysis systems: A discussion report. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA '18)*. Association for Computing Machinery, 2018.

[52] R. Bauer, Q. Q. Ngo, G. Reina, S. Frey, B. Flemisch, H. Hauser, T. Ertl, and M. Sedlmair. Visual ensemble analysis of fluid flow in porous media across simulation codes and experiment. *Transport in Porous Media*, 151(5): 1003–1031, 2024.

[53] E. Beauxis-Aussalet, M. Behrisch, R. Borgo, D. H. Chau, C. Collins, D. Ebert, M. El-Assady, A. Endert, D. A. Keim, J. Kohlhammer, D. Oelke, J. Peltonen, M. Riveiro, T. Schreck, H. Strobelt, and J. J. van Wijk. The role of interactive visualization in fostering trust in AI. *IEEE Computer Graphics and Applications*, 41(6):7–12, 2021.

[54] M. Bellet, J. Bellet, H. Nienborg, Z. M. Hafed, and P. Berens. Human-level saccade detection performance using deep neural networks. *Journal of Neurophysiology*, 2018.

[55] S. Benedetto, M. Pedrotti, and B. Bridgeman. Microsaccades and exploratory saccades in a naturalistic environment. *Journal of Eye Movement Research*, 4(2), 2011.

[56] J. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979.

[57] J. Bernard, N. Wilhelm, M. Scherer, T. May, and T. Schreck. TimeSeriesPaths: Projection-based explorative analysis of multivariate time series data. *Journal of WSCG*, 20(2):97–106, 2012.

[58] G. Bierman, M. Abadi, and M. Torgersen. Understanding TypeScript. In *European Conference on Object-Oriented Programming*, pages 257–281. Springer, 2014.

[59] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc., 2009.

[60] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl. Visualization of eye tracking data: A taxonomy and survey. *Computer Graphics Forum*, 36(8):260–284, 2017.

[61] P. Blignaut. Fixation identification: The optimum threshold for a dispersion algorithm. *Attention, Perception, & Psychophysics*, 71(4):881–895, 2009.

[62] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri. Findings of the 2016 conference on machine translation (WMT16). In *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 131–198. Association for Computational Linguistics, 2016.

[63] Y. S. Bonneh, T. H. Donner, D. Sagi, M. Fried, A. Cooperman, D. J. Heeger, and A. Arieli. Motion-induced blindness and microsaccades: Cause and effect. *Journal of Vision*, article 22, 10(14):1–15, 2010.

[64] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Science & Business Media, 2005.

[65] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[66] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[67] N. Brich, C. Schulz, J. Peter, W. Klingert, M. Schenk, D. Weiskopf, and M. Krone. Visual analysis of multivariate intensive care surveillance data. In *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 71–83. The Eurographics Association, 2020.

[68] N. Brich, C. Schulz, J. Peter, W. Klingert, M. Schenk, D. Weiskopf, and M. Krone. Visual analytics of multivariate intensive care time series data. *Computer Graphics Forum*, 41(6):273–286, 2022.

[69] K. Brodlie, R. Allendes Osorio, and A. Lopes. A review of uncertainty in data visualization. *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109, 2012.

[70] S. Brody, U. Alon, and E. Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

[71] J. Brooke et al. SUS - A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194):4–7, 1996.

[72] E. T. Brown, S. Yarlagadda, K. A. Cook, R. Chang, and A. Endert. ModelSpace: Visualizing the trails of data models in visual analytics systems. In *Proceedings of the Machine Learning from User Interaction for Visualization and Analytics Workshop at IEEE VIS*, 2018.

[73] M. Burch and K. Kurzhals. Visual analysis of eye movements during game play. In *ACM Symposium on Eye Tracking Research and Applications*, article 59, pages 1–5, 2020.

[74] M. Burch, L. Chuang, A. Duchowski, W. Daniel, and R. Groner. Eye tracking and visualization. Introduction to the special thematic issue. *Journal of Eye Movement Research*, 10(5), 2018.

[75] E. Cakmak, D. Seebacher, J. Buchmüller, and D. A. Keim. Time series projection to highlight trends and outliers. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 104–105. IEEE, 2018.

[76] S. K. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[77] S. Carpendale. Evaluating information visualizations. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization: Human-Centered Issues and Perspectives*, pages 19–45. Springer Berlin Heidelberg, 2008.

[78] D. Cashman, G. Patterson, A. Mosca, and R. Chang. RNNbow: Visualizing learning via backpropagation gradients in recurrent neural networks. In *Workshop on Visual Analytics for Deep Learning (VADL)*, 2017.

[79] S. Cha and S. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(06):1355–1370, 2002.

[80] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(4):300–307, 2007.

[81] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26, 2023.

[82] N. Charness, E. M. Reingold, M. Pomplun, and D. M. Stampe. The perceptual aspect of skilled performance in chess: Evidence from eye movements. *Memory & Cognition*, 29(8):1146–1152, 2001.

[83] A. Chatzimparmpas, R. M. Martins, and A. Kerren. t-viSNE: Interactive assessment and interpretation of t-SNE projections. *IEEE Transactions on Visualization and Computer Graphics*, 26(8):2696–2714, 2020.

[84] A. Chatzimparmpas, K. Kucher, and A. Kerren. Visualization for trust in machine learning revisited: The state of the field in 2023. *IEEE Computer Graphics and Applications*, 44(03):99–113, 2024.

[85] H. Chen, G. Wang, D. Peng, W. Zuo, and W. Chen. Sequential document visualization based on hierarchical parametric histogram curves. *Tsinghua Science and Technology*, 17(4):409–418, 2012.

[86] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

[87] G. Choi and M. Kim. Eye gaze information of player using objects in FPS game space. In *Proceedings of the IEEE Global Conference on Consumer Electronics*, pages 1–3, 2017.

[88] F. Chollet. Keras, 2015, https://github.com/fchollet/keras.

[89] J. Choo and S. Liu. Visual analytics for explainable deep learning. *IEEE Computer Graphics and Applications*, 38(4):84–92, 2018.

[90] C. Chu and R. Wang. A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319. Association for Computational Linguistics, 2018.

[91] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286. Association for Computational Linguistics, 2019.

[92] C. Collins, S. Carpendale, and G. Penn. Visualization of uncertainty in lattices to support decision-making. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization (EuroVis 2007)*, pages 51–58, 2007.

[93] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995.

[94] F. M. Costela, M. B. McCamy, S. L. Macknik, J. Otero-Millan, and S. Martinez-Conde. Microsaccades restore the visibility of minute foveal targets. *PeerJ*, 1:e119, 2013.

[95] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[96] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[97] V. Damodaran, S. Chakravarthy, A. Kumar, A. Umapathy, T. Mitamura, Y. Nakashima, N. Garcia, and C. Chu. Understanding the role of scene graphs in visual question answering. arXiv preprint arXiv:2101.05479, 2021.

[98] T. Dang and N. V. T. Nguyen. MultiProjector: Temporal projection for multivariates time series. In G. Bebis, B. Li, A. Yao, Y. Liu, Y. Duan, M. Lau, R. Khadka, A. Crisan, and R. Chang, editors, *Advances in Visual Computing*, pages 91–102. Springer International Publishing, 2022.

[99] S. D'Angelo and D. Gergle. Gazed and confused: Understanding and designing shared gaze for remote collaboration. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2492–2496, 2016.

[100] J. de Leeuw and P. Mair. Multidimensional scaling using majorization: SMACOF in R. *Journal of Statistical Software, Articles*, 31(3):1–30, 2009.

[101] V. De Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

[102] M. Denkowski and G. Neubig. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27. Association for Computational Linguistics, 2017.

[103] H. Deubel and B. Bridgeman. Perceptual consequences of ocular lens overshoot during saccadic eye movements. *Vision Research*, 35(20):2897–2902, 1995.

[104] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics, 2019.

[105] L. L. Di Stasi, M. B. McCamy, A. Catena, S. L. Macknik, J. J. Cañas, and S. Martinez-Conde. Microsaccade and drift dynamics reflect mental fatigue. *European Journal of Neuroscience*, 38(3):2389–2398, 2013.

[106] O. Dimigen, M. Valsecchi, W. Sommer, and R. Kliegl. Human microsaccade-related visual brain responses. *Journal of Neuroscience*, 29 (39):12321–12331, 2009.

[107] D. L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. In *AMS Conference on Math Challenges of the 21st Century*, 2000.

[108] A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer, 2007.

[109] O. Dušek, J. Hajič, J. Hlaváčová, J. Libovický, P. Pecina, A. Tamchyna, and Z. Urešová. Khresmoi summary translation test data 2.0, 2017, Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics (UFAL).

[110] M. S. El-Nasr, A. Drachen, and A. Canossa, editors. *Game Analytics*. Springer, 2016.

[111] R. Engbert and R. Kliegl. Microsaccades uncover the orientation of covert attention. *Vision Research*, 43(9):1035–1045, 2003.

[112] R. Engbert and R. Kliegl. Chapter 6 - Binocular coordination in microsaccades. In J. Hyönä, R. Radach, and H. Deubel, editors, *The Mind's Eye*, pages 103–117. North-Holland, 2003.

[113] R. Engbert and K. Mergenthaler. Microsaccades are triggered by low retinal image slip. *Proceedings of the National Academy of Sciences*, 103(18): 7192–7197, 2006.

[114] R. Engbert, P. Sinn, K. Mergenthaler, and H. Trukenbrod. Microsaccade toolbox. http://read.psych.uni-potsdam.de/attachments/article/140/MS_Toolbox_R.zip, 2015.

[115] D. Engel, L. Hüttenberger, and B. Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In C. Garth, A. Middel, and H. Hagen, editors, *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*, volume 27 of *OASICS*, pages 135–149. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

[116] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, 2009.

[117] K. A. Ericsson and H. A. Simon. *Protocol Analysis*. MIT Press Cambridge, 1993.

[118] M. Espadoto, R. M. Martins, A. Kerren, N. S. T. Hirata, and A. C. Telea. Toward a quantitative survey of dimension reduction rechniques. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2153–2173, 2021.

[119] M. H. Everts, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1299–1306, 2009.

[120] M. C. Fahey, P. D. Cremer, S. T. Aw, L. Millist, M. J. Todd, O. B. White, M. Halmagyi, L. A. Corben, V. Collins, A. J. Churchyard, et al. Vestibular, saccadic and fixation abnormalities in genetically confirmed friedreich ataxia. *Brain*, 131(4):1035–1045, 2008.

[121] Y. Fang, H. Xu, and J. Jiang. A survey of time series data visualization research. *IOP Conference Series: Materials Science and Engineering*, 782(2): 022013, 2020.

[122] R. Faust, D. Glickenstein, and C. Scheidegger. DimReader: Axis lines that explain non-linear projections. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):481–490, 2018.

[123] S. J. Fernstad, J. Shaw, and J. Johansson. Quality-based guidance for exploratory dimensionality reduction. *Information Visualization*, 12(1): 44–64, 2013.

[124] C. Forsell and M. Cooper. An introduction and guide to evaluation of visualization techniques through user studies. In W. Huang, editor, *Handbook of Human Centric Visualization*, pages 285–313. Springer New York, 2014.

[125] A. G. Foundation. KGS-Go-Server. https://www.gokgs.com/, 2019, [Online; accessed 06-December-2019].

[126] M. Fried, E. Tsitsiashvili, Y. S. Bonneh, A. Sterkin, T. Wygnanski-Jaffe, T. Epstein, and U. Polat. ADHD subjects fail to suppress eye blinks and microsaccades while anticipating visual stimuli but recover with medication. *Vision Research*, 101(Supplement C):62–72, 2014.

[127] Y. Fu and J. T. Stasko. HoopInSight: Analyzing and comparing basketball shooting performance through visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):858–868, 2024.

[128] Y. Gao and B. A. Sabel. Microsaccade dysfunction and adaptation in hemianopia after stroke. *Restorative Neurology and Neuroscience*, 35(4): 365–376, 2017.

[129] R. Garcia and D. Weiskopf. Inner-process visualization of hidden states in recurrent neural networks. In *Proceedings of the 13th International Symposium on Visual Information Communication and Interaction (VINCI '20)*, article 20, pages 1–5, 2020.

[130] R. Garcia, A. C. Telea, B. C. da Silva, J. Tørresen, and J. L. D. Comba. A task-and-technique centered survey on visual analytics for deep learning model engineering. *Computers & Graphics*, 77:30–49, 2018.

[131] H. Gharaee, S. Z. Ghanavati, S. S. Rad, A. Omidtabrizi, and H. Naseri. Effectiveness of Technolas torsional eye tracking system on visual outcomes after photorefractive keratectomy. *Journal of Current Ophthalmology*, 27(3): 82–86, 2015.

[132] S. Ghosh, G. Burachas, A. Ray, and A. Ziskind. Generating natural language explanations for visual question answering using scene graphs and visual attention. arXiv preprint arXiv:1902.05715, 2019.

[133] Y. Goyal, A. Mohapatra, D. Parikh, and D. Batra. Towards transparent AI systems: Interpreting visual question answering models. arXiv preprint arXiv:1608.08974, 2016.

[134] A. Gracia, S. González, V. Robles, and E. Menasalvas. A methodology to compare dimensionality reduction algorithms in terms of loss of quality. *Information Sciences*, 270:1–27, 2014.

[135] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[136] S. Green, J. Heer, and C. D. Manning. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, pages 439–448. ACM, 2013.

[137] S. Green, J. Heer, and C. D. Manning. Natural language translation at the intersection of AI and HCI. *Communications of the ACM*, 58(9):46–53, 2015.

[138] M. R. Greene, T. Liu, and J. M. Wolfe. Reconsidering Yarbus: A failure to predict observers' task from eye movement patterns. *Vision Research*, 62: 1–8, 2012.

[139] M. Grinberg. *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc., 2018.

[140] Y. Guo, S. Guo, Z. Jin, S. Kaul, D. Gotz, and N. Cao. Survey on visual analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):5091–5112, 2022.

[141] Z. M. Hafed and J. J. Clark. Microsaccades as an overt measure of covert attention shifts. *Vision Research*, 42(22):2533–2545, 2002.

[142] D. Hägele, M. Abdelaal, O. S. Oguz, M. Toussaint, and D. Weiskopf. Visualization of nonlinear programming for robot motion planning. In *Proceedings of the 13th International Symposium on Visual Information Communication and Interaction (VINCI '20)*, article 10, pages 1–8. Association for Computing Machinery, 2020.

[143] D. Hägele, M. Abdelaal, O. S. Oguz, M. Toussaint, and D. Weiskopf. Visual analytics for nonlinear programming in robot motion planning. *Journal of Visualization*, 25:127–141, 2022.

[144] A. Hassan. Visualization of neural networks for diagnostic hydrological modeling, 2022, Master's thesis, University of Stuttgart, doi: 10.18419/opus-13026.

[145] S. Havre, B. Hetzler, and L. Nowell. ThemeRiver: Visualizing theme changes over time. In *IEEE Symposium on Information Visualization*, pages 115–123. IEEE, 2000.

[146] J. He and C. Chen. Visualizing temporal patterns in representation data. In *VADL 2017: Workshop on Visual Analytics for Deep Learning*, volume 4, 2017.

[147] J. Heer, N. Kong, and M. Agrawala. Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1303–1312, 2009.

[148] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015.

[149] F. Hermens and R. Walker. What determines the direction of microsaccades? *Journal of Eye Movement Research*, 3(4), 2010.

[150] R. S. Hessels, G. A. Holleman, T. H. W. Cornelissen, I. T. C. Hooge, and C. Kemner. Eye contact takes two – autistic and social anxiety traits predict gaze behavior in dyadic interaction. *Journal of Experimental Psychopathology*, 9(2):jep.062917, 2018.

[151] R. S. Hessels, G. A. Holleman, A. Kingstone, I. T. Hooge, and C. Kemner. Gaze allocation in face-to-face communication is affected primarily by task structure and social context, not stimulus-driven factors. *Cognition*, 184:28–43, 2019.

[152] N. Heulot, M. Aupetit, and J.-D. Fekete. Poster: Proxiviz: An interactive visualization technique to overcome multidimensional scaling artifacts. *Proceedings of IEEE InfoVis*, 2, 2012.

[153] F. Heyen. Visual parameter space analysis for classification models, 2019, Master's thesis, University of Stuttgart, doi: 10.18419/opus-10728.

[154] H. Hicheur, S. Zozor, A. Campagne, and A. Chauvin. Microsaccades are modulated by both attentional demands of a visual discrimination task and background noise. *Journal of Vision*, 13(13):18, 2013.

[155] A. Hinterreiter, C. Steinparz, M. Schöfl, H. Stitz, and M. Streit. Projection path explorer: Exploring visual patterns in projected decision-making paths. *ACM Transactions on Interactive Intelligent Systems*, 11(3–4):1–29, 2021.

[156] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[157] F. M. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2674–2693, 2018.

[158] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer. *Eye Tracking: A Comprehensive Guide to Methods and Measures*. Oxford University Press, 2011.

[159] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spaCy: Industrial-strength natural language processing in Python. Zenodo, 2020.

[160] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

[161] P.-J. Hsieh and P. U. Tse. Microsaccade rate varies with subjective visibility during motion-induced blindness. *PLOS ONE*, 4(4):1–9, 2009.

[162] Y. Hu, S. Wu, S. Xia, J. Fu, and W. Chen. Motion Track: Visualizing variations of human motion data. In *2010 IEEE Pacific Visualization Symposium*, pages 153–160, 2010.

[163] H. Huang, X. Hu, Y. Zhao, M. Makkie, Q. Dong, S. Zhao, L. Guo, and T. Liu. Modeling task fMRI data via deep convolutional autoencoder. *IEEE Transactions on Medical Imaging*, 37(7):1551–1561, 2017.

[164] Z. Huang, Z. Zeng, B. Liu, D. Fu, and J. Fu. Pixel-BERT: Aligning image pixels with text by deep multi-modal transformers. arXiv preprint arXiv:2004.00849, 2020.

[165] D. A. Hudson and C. D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6693–6702. IEEE Computer Society, 2019.

[166] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1 (2):69–91, 1985.

[167] T. Isenberg, P. Isenberg, J. Chen, M. Sedlmair, and T. Möller. A systematic review on the practice of evaluating visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2818–2827, 2013.

[168] A. Ishida and J. Davies. *Attack and Defense*. The Ishi Press, 1979.

[169] M. F. Ishmam, M. S. H. Shovon, M. Mridha, and N. Dey. From image to language: A critical analysis of visual question answering (VQA) approaches, challenges, and opportunities. *Information Fusion*, 106:102270, 2024.

[170] P. Isokoski, M. Joos, O. Spakov, and B. Martin. Gaze controlled games. *Universal Access in the Information Society*, 8(4):323–337, 2009.

[171] N. Jain, A. Bhansali, and D. Mehta. AngularJS: A modern MVC framework in JavaScript. *Journal of Global Research in Computer Science*, 5(12):17–23, 2014.

[172] R. Jasiek. *Tactical Reading*. R. Jasiek, 2015.

[173] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *2012 IEEE Pacific Visualization Symposium (PacificVis)*, pages 1–8, 2012.

[174] P. Jermann, M. Nüssli, and W. Li. Using dual eye-tracking to unveil coordination and expertise in collaborative tetris. In *Proceedings of the 24th BCS Interaction Specialist Group Conference*, pages 36–44. BCS Learning & Development Ltd., 2010.

[175] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3668–3678. Institute of Electrical and Electronics Engineers (IEEE), 2015.

[176] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato. Local affine multidimensional projection. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2563–2571, 2011.

[177] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97, 2017.

[178] A. Karpathy, J. Johnson, and F. Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[179] R. Karsh and F. Breitenbach. Looking at looking: The amorphous fixation measure. *Eye Movements and Psychological Functions: International Views*, pages 53–64, 1983.

[180] S. Kaski and J. Peltonen. Dimensionality reduction for data visualization. *IEEE Signal Processing Magazine*, 28(2):100–104, 2011.

[181] M. Kaya. System zur visuellen Analyse for Daten der kontinuierlichen Glukosemessung von Diabetes-Patienten, 2021, Bachelor's thesis, University of Stuttgart, doi: 10.18419/opus-11935.

[182] D. A. Keim, F. Mansmann, J. Schneidewind, J. J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*, pages 76–90. Springer, 2008.

[183] P. Koehn and R. Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics, 2017.

[184] T. Kohonen. *Self-Organizing Maps*, volume 30. Springer Science & Business Media, 2012.

[185] K. Krejtz, A. T. Duchowski, A. Niedzielska, C. Biele, and I. Krejtz. Eye tracking cognitive load using pupil diameter and microsaccades with fixed gaze. *PLOS ONE*, 13(9):e0203629, 2018.

[186] B. Krekelberg. Microsaccades. *Current Biology*, 21(11):R416, 2011.

[187] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[188] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[189] K. Kucher and A. Kerren. Text visualization techniques: Taxonomy, visual survey, and community insights. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pages 117–121, 2015.

[190] A. Kumar, M. Burch, and K. Mueller. Visual analysis of eye gazes to assist strategic planning in computer games. In *Proceedings of the 3rd Workshop on Eye Tracking and Visualization*, article 6, pages 1–5, 2018.

[191] A. Kumar, P. Howlader, R. Garcia, D. Weiskopf, and K. Mueller. Challenges in interpretability of neural networks for eye movement data. In *ACM Symposium on Eye Tracking Research and Applications (ETRA '20 Short Papers)*, article 12, pages 1–5, 2020.

[192] K. Kurzhals and D. Weiskopf. Space-time visual analytics of eye-tracking data for dynamic stimuli. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2129–2138, 2013.

[193] K. Kurzhals, B. Fisher, M. Burch, and D. Weiskopf. Evaluating visual analytics with eye tracking. In *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization (BELIV '14)*, page 61–69. Association for Computing Machinery, 2014.

[194] K. Kurzhals, M. Hlawatsch, F. Heimerl, M. Burch, T. Ertl, and D. Weiskopf. Gaze Stripes: Image-based visualization of eye tracking data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1005–1014, 2016.

[195] K. Kurzhals, M. Burch, T. Blascheck, G. Andrienko, N. Andrienko, and D. Weiskopf. A task-based view on the visual analysis of eye tracking data. In M. Burch, L. Chuang, B. Fisher, A. Schmidt, and D. Weiskopf, editors, *Eye Tracking and Visualization*, pages 3–22. Springer, 2017.

[196] P. Kuznecov. A visual analytics approach for explainability of deep neural networks, 2018, Master's thesis, University of Stuttgart, doi: 10.18419/opus-10151.

[197] B. La Rosa, G. Blasilli, R. Bourqui, D. Auber, G. Santucci, R. Capobianco, E. Bertini, R. Giot, and M. Angelini. State of the art of visual analytics for explainable deep learning. *Computer Graphics Forum*, 42(1):319–355, 2023.

[198] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1520–1536, 2012.

[199] M. Lankes, B. Maurer, and B. Stiglbauer. An eye for an eye: Gaze input in competitive online games and its effects on social presence. In *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology*, article 17, pages 1–9, 2016.

[200] J. Laubrock, R. Engbert, and R. Kliegl. Microsaccade dynamics during covert attention. *Vision Research*, 45(6):721–730, 2005.

[201] J. Laubrock, R. Kliegl, M. Rolfs, and R. Engbert. When do microsaccades follow spatial attention? *Attention, Perception, & Psychophysics*, 72(3): 683–694, 2010.

[202] J. Lee, J.-H. Shin, and J.-S. Kim. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126. Association for Computational Linguistics, 2017.

[203] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[204] S. Lespinats and M. Aupetit. CheckViz: Sanity check and topological clues for linear and non-linear mappings. *Computer Graphics Forum*, 30(1): 113–125, 2011.

[205] X. Li, Q.and Tang and Y. Jian. Adversarial learning with bidirectional attention for visual question answering. *Sensors*, 21(21):7164, 2021.

[206] W. Liang, Y. Jiang, and Z. Liu. GraphVQA: Language-guided graph neural networks for graph-based visual question answering. In *Proceedings of the Third Workshop on Multimodal Artificial Intelligence*, pages 79–86. Association for Computational Linguistics, 2021.

[207] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

[208] S. Liu, B. Wang, P.-T. Bremer, and V. Pascucci. Distortion-guided structure-driven interactive exploration of high-dimensional data. *Computer Graphics Forum*, 33(3):101–110, 2014.

[209] S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1): 48–56, 2017.

[210] S. Liversedge, I. Gilchrist, and S. Everling. *The Oxford handbook of eye movements*. Oxford University Press, 2011.

[211] E. Lowet, B. Gomes, K. Srinivasan, H. Zhou, R. J. Schafer, and R. Desimone. Enhanced neural processing by covert attention only during microsaccades directed toward the attended stimulus. *Neuron*, 2018.

[212] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics, 2015.

[213] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11)*, pages 142–150, 2011.

[214] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan. Visual semiotics & uncertainty visualization: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2496–2505, 2012.

[215] I. V. Malinov, J. Epelboim, A. N. Herst, and R. M. Steinman. Characteristics of saccades and vergence in two kinds of sequential looking tasks. *Vision Research*, 40(16):2083–2090, 2000.

[216] Y. Mao, J. Dillon, and G. Lebanon. Sequential document visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1208–1215, 2007.

[217] S. Martinez-Conde, S. L. Macknik, and D. H. Hubel. The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*, 5(3): 229–240, 2004.

[218] S. Martinez-Conde, S. L. Macknik, X. G. Troncoso, and D. H. Hubel. Microsaccades: A neurophysiological analysis. *Trends in Neurosciences*, 32 (9):463–475, 2009.

[219] R. M. Martins, D. B. Coimbra, R. Minghim, and A. C. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Computers & Graphics*, 41:26–42, 2014.

[220] B. Maurer, M. Lankes, B. Stiglbauer, and M. Tscheligi. EyeCo: Effects of shared gaze on social presence in an online cooperative game. In *International Conference on Entertainment Computing*, pages 102–114, 2016.

[221] M. B. McCamy, N. Collins, J. Otero-Millan, M. Al-Kalbani, S. L. Macknik, D. Coakley, X. G. Troncoso, G. Boyle, V. Narayanan, T. R. Wolf, and S. Martinez-Conde. Simultaneous recordings of ocular microtremor and microsaccades with a piezoelectric sensor and a video-oculography system. *PeerJ*, 1:e14, 2013.

[222] M. B. McCamy, A. Najafian Jazi, J. Otero-Millan, S. L. Macknik, and S. Martinez-Conde. The effects of fixation target size and luminance on microsaccades and square-wave jerks. *PeerJ*, 1:e9, 2013.

[223] M. B. McCamy, J. Otero-Millan, L. L. Di Stasi, S. L. Macknik, and S. Martinez-Conde. Highly informative natural scene regions increase microsaccade production during visual scanning. *Journal of Neuroscience*, 34(8):2956–2966, 2014.

[224] M. B. McCamy, J. Otero-Millan, R. J. Leigh, S. A. King, R. M. Schneider, S. L. Macknik, and S. Martinez-Conde. Simultaneous recordings of human microsaccades and drifts with a contemporary video eye tracker and the search coil technique. *PLOS ONE*, 10(6):1–20, 2015.

[225] L. McInnes, J. Healy, N. Saul, and L. Grossberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3: 861, 2018.

[226] K. Mergenthaler and R. Engbert. Microsaccades are different from saccades in scene perception. *Experimental Brain Research*, 203(4):753–757, 2010.

[227] A. Mihali, B. van Opheusden, and W. J. Ma. Bayesian microsaccade detection. *Journal of Vision*, 17(1):13, 2017.

[228] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *Proceedings of the 2017 IEEE Conference on Visual Analytics Science and Technology*, pages 13–24, 2017.

[229] B. Mokbel, W. Lueks, A. Gisbrecht, and B. Hammer. Visualizing the quality of dimensionality reduction. *Neurocomputing*, 112:109–123, 2013, Advances in artificial neural networks, machine learning, and computational intelligence.

[230] V. Molchanov and L. Linsen. Visual exploration of patterns in multi-run time-varying multi-field simulation data using projected views. In *21st International Conference in Central Europeon Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS Association (WSCG 2014)*, pages 39–48. Václav Skala - UNION Agency, 2014.

[231] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009.

[232] J. Newn, E. Velloso, F. Allison, Y. Abdelrahman, and F. Vetere. Evaluating real-time gaze representations to infer intentions in competitive turn-based strategy games. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, pages 541–552, 2017.

[233] J. Newn, F. Allison, E. Velloso, and F. Vetere. Looks can be deceiving: Using gaze visualisation to predict and mislead opponents in strategic gameplay. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, article 261, pages 1–12, 2018.

[234] H. D. Nguyen. Visual exploration for deep learning models and trainings for microstructure data, 2022, Master's thesis, University of Stuttgart, doi: 10.18419/opus-13025.

[235] D. C. Niehorster, T. Cornelissen, K. Holmqvist, and I. Hooge. Searching with and against each other: Spatiotemporal coordination of visual search behavior in collaborative and competitive settings. *Attention, Perception, & Psychophysics*, 81(3):666–683, 2019.

[236] L. G. Nonato and M. Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2650–2673, 2019.

[237] W. Norcliffe-Brown, S. Vafeias, and S. Parisot. Learning conditioned graph structures for interpretable visual question answering. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[238] D. Noton and L. Stark. Scanpaths in saccadic eye movements while viewing and recognizing patterns. *Vision Research*, 11(9):929–942, 1971.

[239] D. Noton and L. Stark. Scanpaths in eye movements during pattern perception. *Science*, 171(3968):308–311, 1971.

[240] OpenCV Dev Team. Histogram comparison — OpenCV 2.4.13.7 documentation, 2011–2014. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html, Accessed 06.08.2019.

[241] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, J. Akinjobi, et al. Supervised machine learning algorithms: Classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3):128–138, 2017.

[242] J. Otero-Millan, X. G. Troncoso, S. L. Macknik, I. Serrano-Pedraza, and S. Martinez-Conde. Saccades and microsaccades during visual fixation, exploration, and search: Foundations for a common saccadic generator. *Journal of Vision*, 8(14):1–18, 2008.

[243] J. Otero-Millan, S. L. Macknik, and S. Martinez-Conde. Microsaccades and blinks trigger illusory rotation in the "rotating snakes" illusion. *Journal of Neuroscience*, 32(17):6043–6051, 2012.

[244] J. Otero-Millan, J. L. A. Castro, S. L. Macknik, and S. Martinez-Conde. Unsupervised clustering method to detect microsaccades. *Journal of Vision*, 14(2):18, 2014.

[245] A. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997.

[246] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02)*, pages 311–318, 2002.

[247] A. Pastukhov and J. Braun. Rare but precious: Microsaccades are highly informative about attentional allocation. *Vision Research*, 50(12):1173–1184, 2010.

[248] A. Pastukhov, V. Vonau, S. Stonkute, and J. Braun. Spatial and temporal attention revealed by microsaccades. *Vision Research*, Visual Attention 2013 Volume II, 85(Supplement C):45–57, 2013.

[249] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[250] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[251] C. Perin, T. Wun, R. Pusch, and S. Carpendale. Assessing the graphical perception of time and speed on 2D+ time trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):698–708, 2017.

[252] Á. Peris, L. Cebrián, and F. Casacuberta. Online learning for neural machine translation post-editing. *arXiv preprint arXiv:1706.03196*, 2017.

[253] A. Piras, M. Raffi, M. Perazzolo, I. M. Lanzoni, and S. Squatrito. Microsaccades and interest areas during free-viewing sport task. *Journal of Sports Sciences*, 37(9):980–987, 2019.

[254] M. Poletti and M. Rucci. A compact field guide to the study of microsaccades: Challenges and functions. *Vision Research*, 118(Supplement C): 83–97, 2016.

[255] M. Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics, 2018.

[256] K. Potter, P. Rosen, and C. R. Johnson. From quantification to visualization: A taxonomy of uncertainty visualization approaches. In A. M. Dienstfrey and R. F. Boisvert, editors, *Uncertainty Quantification in Scientific Computing*, pages 226–249. Springer, 2012.

[257] L. Prandtl, O. G. Tietjens, and J. P. Den Hartog. *Applied hydro- and aeromechanics*. Dover Publications, 2003.

[258] C. M. Privitera, T. Carney, S. Klein, and M. Aguilar. Analysis of microsaccades and pupil dilation reveals a common decisional origin during visual search. *Vision Research*, 95(Supplement C):43–50, 2014.

[259] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[260] M. Quist and G. Yona. Distributional scaling: An algorithm for structure-preserving embedding of metric and nonmetric spaces. *The Journal of Machine Learning Research*, 5:399–420, 2004.

[261] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[262] M. Raj and R. T. Whitaker. Visualizing multidimensional data with order statistics. *Computer Graphics Forum*, 37(3):277–287, 2018.

[263] N. F. Rajani and R. J. Mooney. Ensembling visual explanations for VQA. In *Proceedings of the NIPS 2017 Workshop on Visually-Grounded Interaction and Language*, 2017.

[264] P. E. Rauber, A. X. Falcão, and A. C. Telea. Visualizing time-dependent data using dynamic t-SNE. In *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers (EuroVis '16)*, pages 73–77. Eurographics Association, 2016.

[265] P. E. Rauber, S. G. Fadel, A. X. Falcão, and A. C. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):101–110, 2017.

[266] A. Ray, M. Cogswell, X. Lin, K. Alipour, A. Divakaran, Y. Yao, and G. Burachas. Generating and evaluating explanations of attended and error-inducing input regions for VQA models. *Applied AI Letters*, 2(4):e51, 2021.

[267] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, 1981.

[268] Reuters Ltd. Reuters-21578 dataset, 1996. [Online]. Available: http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.

[269] G. Richer, A. Pister, M. Abdelaal, J.-D. Fekete, M. Sedlmair, and D. Weiskopf. Scalability in visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):3314–3330, 2024.

[270] B. Rieck and H. Leitte. Persistent homology for the evaluation of dimensionality reduction schemes. *Computer Graphics Forum*, 34(3):431–440, 2015.

[271] M. Rikters. Debugging neural machine translations. arXiv preprint arXiv:1808.02733, 2018.

[272] M. Rikters, M. Fishel, and O. Bojar. Visualizing neural machine translation attention and confidence. *The Prague Bulletin of Mathematical Linguistics*, 109(1):39–50, 2017.

[273] J. C. Roberts. State of the art: Coordinated multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*, pages 61–71, 2007.

[274] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[275] S. R. Salian. Deep visualization for MR-based biological age estimation, 2020, Master's thesis, University of Stuttgart, doi: 10.18419/opus-11386.

[276] 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 (ETRA '00)*, pages 71–78. Association for Computing Machinery, 2000.

[277] D. B. S. Satoaki Eitschberger, Pascal Walloner. A curated collection of multivariate time series for visualization benchmarks, 2021, Bachelor-Forschungsprojekt, University of Stuttgart.

[278] T. Schreck, T. Tekušová, J. Kohlhammer, and D. Fellner. Trajectory-based visual analysis of large financial time series data. *ACM SIGKDD Explorations Newsletter*, 9(2):30–37, 2007.

[279] N. Schäfer. Visuelle Eye-Tracking-Analyse bei kooperativen Spielszenarien, 2019, Bachelor's thesis, University of Stuttgart, doi: 10.18419/opus-11411.

[280] N. Schäfer. Visual Analytics für Visual-Reasoning-Aufgaben, 2022, Master's thesis, University of Stuttgart, doi: 10.18419/opus-12677.

[281] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012.

[282] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368, 2017.

[283] C. Seifert, V. Sabol, and W. Kienreich. Stress Maps: Analysing local phenomena in dimensionality reduction based visualisations. In *EuroVAST@ EuroVis*, 2010.

[284] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics, 2016.

[285] R. Sennrich, B. Haddow, and A. Birch. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016*, pages 371–376. The Association for Computer Linguistics, 2016.

[286] Sensei's Library contributors. Pair Go, 2019. [Online]. Available: https://senseis.xmp.net/?Rengo, Accessed 25.03.2020.

[287] A. Shvarts, A. Stepanov, and D. Chumachenko. Automatic detection of gaze convergence in multimodal collaboration: A dual eye-tracking technology. *The Russian Journal of Cognitive Science*, 5(3):4–17, 2018.

[288] D. Silver, A. Huang, C. J. Maddison, A. Guez, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[289] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, et al. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, 2017.

[290] P. Sinn and R. Engbert. Saccadic facilitation by modulation of microsaccades in natural backgrounds. *Attention, Perception, & Psychophysics*, 73(4): 1029–1033, 2011.

[291] P. Sinn and R. Engbert. Small saccades versus microsaccades: Experimental distinction and model-based unification. *Vision Research*, 118 (Supplement C):132–143, 2016.

[292] D. Smith and T. C. N. Graham. Use of eye movements for video game control. In *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pages 20–28, 2006.

[293] O. Špakov. EyeChess: The tutoring game with visual attentive interface. *Alternative Access: Feelings & Games*, 5, 2005.

[294] O. Špakov and D. Miniotas. Visualization of eye gaze data using heat maps. *Electronics and Electrical Engineering*, 2(74):55–58, 2007.

[295] O. Špakov, H. Istance, K. Räihä, T. Viitanen, and H. Siirtola. Eye gaze and head gaze in collaborative games. In *Proceedings of the ACM Symposium on Eye Tracking Research & Applications*, article 85, pages 1–9, 2019.

[296] O. Špakov, D. Niehorster, H. Istance, K. Räihä, and H. Siirtola. Two-way gaze sharing in remote teaching. In D. Lamas, F. Loizides, L. Nacke, H. Petrie, M. Winckler, and P. Zaphiris, editors, *Human-Computer Interaction (INTERACT 2019)*, pages 242–251. Springer International Publishing, 2019.

[297] F. Stahlberg. Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418, 2020.

[298] J. Stahnke, M. Dörk, B. Müller, and A. Thom. Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE Transactions on Visualization and Computer Graphics*, 22(1): 629–638, 2015.

[299] C. A. Steinparz, A. Hinterreiter, H. Stitz, and M. Streit. Visualization of rubik's cube solution algorithms. In T. von Landesberger and C. Turkay, editors, *EuroVis Workshop on Visual Analytics (EuroVA '19)*. The Eurographics Association, 2019.

[300] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018.

[301] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush. Seq2Seq-Vis: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1): 353–363, 2019.

[302] S. Strunge Mathiesen and H.-J. Schulz. Aesthetics and ordering in stacked area charts. In A. Basu, G. Stapleton, S. Linker, C. Legg, E. Manalo, and P. Viana, editors, *Diagrammatic Representation and Inference*, pages 3–19. Springer International Publishing, 2021.

[303] V. Sundstedt. Gazing at games: An introduction to eye tracking control. *Synthesis Lectures on Computer Graphics and Animation*, 5(1):1–113, 2012.

[304] V. Sundstedt, M. Bernhard, E. Stavrakis, E. Reinhard, and M. Wimmer. Visual attention and gaze behavior in games: An object-based approach. In M. S. El-Nasr, A. Drachen, and A. Canossa, editors, *Game Analytics*, pages 543–583. Springer, 2013.

[305] Z. Tan, S. Wang, Z. Yang, G. Chen, X. Huang, M. Sun, and Y. Liu. Neural machine translation: A review of methods, resources, and tools. *AI Open*, 1:5–21, 2020.

[306] H. Tarner and F. Beck. Visualizing runtime evolution paths in a multi-dimensional space (work in progress paper). In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion)*, pages 33–38. Association for Computing Machinery, 2023.

[307] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500): 2319–2323, 2000.

[308] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center, 2005.

[309] A. Toda, S. Hiwa, K. Tanioka, and T. Hiroyasu. Visualization, clustering, and graph generation of optimization search trajectories for evolutionary computation through topological data dnalysis: Application of the mapper. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2022.

[310] C. Tominski and W. Aigner. The TimeViz Browser – A Visual Survey of Visualization Techniques for Time-Oriented Data. https://browser.timeviz.net, 2023. [Online]. Available: https://browser.timeviz.net, Version 2.0.

[311] C. Tominski, W. Aigner, S. Miksch, and H. Schumann. Images of time. *Information Design: Research and Practice*, pages 23–42, 2017.

[312] W. S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419, 1952.

[313] T. Tsutsumi, Y. Ono, and T. Taketsugu. Visualization of reaction route map and dynamical trajectory in reduced dimension. *Chemical Communications*, 57(89):11734–11750, 2021.

[314] Z. Tu, Y. Liu, L. Shang, X. Liu, and H. Li. Neural machine translation with reconstruction. In *Thirty-First Conference on Artificial Intelligence (AAAI)*, pages 3097–3103, 2017.

[315] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press LLC, 2 edition, 2001.

[316] J. Turner, E. Velloso, H. Gellersen, and V. Sundstedt. EyePlay: Applications for gaze in games. In *Proceedings of the First ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play*, pages 465–468, 2014.

[317] J. Ullerich. Stimulus generation software for the analysis of smooth pursuits, 2019, Bachelor's thesis, University of Stuttgart, doi: 10.18419/opus-10512.

[318] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22 (1):1–10, 2016.

[319] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[320] L. van der Maaten, E. Postma, and H. Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10, 2007.

[321] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, 2009.

[322] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[323] S. Velliangiri, S. Alagumuthukrishnan, et al. A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165:104–111, 2019.

[324] E. F. Vernier, R. Garcia, I. P. d. Silva, J. L. D. Comba, and A. C. Telea. Quantitative evaluation of time-dependent multidimensional projection techniques. *Computer Graphics Forum*, 39(3):241–252, 2020.

[325] R. Vertegaal. The GAZE groupware system: Mediating joint attention in multiparty communication and collaboration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 294–301, 1999.

[326] J. Vig. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42. Association for Computational Linguistics, 2019.

[327] J. Vig. Visualizing attention in transformerbased language models. arXiv preprint arXiv:1904.02679, 2019.

[328] M. H. Vu, T. Löfstedt, T. Nyholm, and R. Sznitman. A question-centric model for visual question answering in medical imaging. *IEEE Transactions on Medical Imaging*, 39(9):2856–2868, 2020.

[329] J. Wang, J. Wang, W. Fang, and H. Niu. Financial time series prediction using Elman recurrent random neural networks. *Computational Intelligence and Neuroscience*, 2016, 2016, article ID 4742515.

[330] J. Wang, S. Liu, and W. Zhang. Visual analytics for machine learning: A data perspective survey. *IEEE Transactions on Visualization and Computer Graphics*, 30(12):7637–7656, 2024.

[331] W. Wang, J.-T. Peter, H. Rosendahl, and H. Ney. CharacTER: Translation edit rate on character level. In *Proceedings of the First Conference on Machine Translation*, volume 2, Shared Task Papers, pages 505–510, 2016.

[332] M. O. Ward. Linking and brushing. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 1623–1626. Springer US, 2009.

[333] M. O. Ward and Z. Guo. Visual exploration of time-series data with shape space projections. *Computer Graphics Forum*, 30(3):701–710, 2011.

[334] M. Wattenberg, F. Viégas, and I. Johnson. How to use t-sne effectively. *Distill*, 2016.

[335] M. Weber, M. Alexa, and W. Muller. Visualizing time-series on spirals. In *IEEE Symposium on Information Visualization 2001 (InfoVis 2001)*, pages 7–13, 2001.

[336] R. B. Weber and R. B. Daroff. Corrective movements following refixation saccades: Type and control system analysis. *Vision Research*, 12(3):467–475, 1972.

[337] N. Weibel, A. Fouse, C. Emmenegger, S. Kimmich, and E. Hutchins. Let's look at the cockpit: Exploring mobile eye-tracking for observational research on the flight deck. In *Proceedings of the ACM Symposium on Eye Tracking Research & Applications*, pages 107–114, 2012.

[338] Wikipedia. Autonomes Fahren — Wikipedia, die freie Enzyklopädie, 2018. [Online]. Available: https://de.wikipedia.org/wiki/Autonomes_Fahren, Accessed 01.03.2019.

[339] Wikipedia. Game complexity — Wikipedia, the free encyclopedia, 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Game_complexity&oldid=916663416, Accessed 20.09.2019.

[340] Wikipedia. Künstliche Intelligenz — Wikipedia, die freie Enzyklopädie, 2021. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Kuenstliche_Intelligenz&oldid=214701516, Accessed 13.08.2021.

[341] L. Wu, P. Cui, J. Pei, L. Zhao, and X. Guo. Graph neural networks: Foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*, pages 4840–4841. Association for Computing Machinery, 2022.

[342] Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40, 2017.

[343] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[344] S. Yang, Y. Wang, and X. Chu. A survey of deep learning techniques for neural machine translation. arXiv preprint arXiv:2002.07526, 2020.

[345] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu. Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528, 2013.

[346] N. Yao, J. Brewer, S. D'Angelo, M. Horn, and D. Gergle. Visualizing gaze information from multiple students to support remote instruction. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems (CHI EA '18)*, pages 1–6, 2018.

[347] A. L. Yarbus. Eye movements during perception of complex objects. In *Eye Movements and Vision*, pages 171–211. Springer US, 1967.

[348] A. J. Yepes, A. Neveol, M. Neves, K. Verspoor, O. Bojar, A. Boyer, C. Grozea, B. Haddow, M. Kittner, Y. Lichtblau, P. Pecina, R. Roller, R. Rosa, A. Siu, P. Thomas, and S. Trescher. Findings of the WMT 2017 biomedical translation shared task. In *Proceedings of the Conference on Machine Translation (WMT)*, volume 2: Shared Task Papers, pages 234–247. Association for Computational Linguistics, 2017.

[349] T. Yokoyama, Y. Noguchi, and S. Kita. Attentional shifts by gaze direction in voluntary orienting: Evidence from a microsaccade study. *Experimental Brain Research*, 223(2):291–300, 2012.

[350] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[351] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. *Computational Visual Media*, pages 3–36, 2020.

[352] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015.

[353] C. Yuksel, S. Schaefer, and J. Keyser. Parameterization and applications of Catmull-Rom curves. *Computer-Aided Design*, 43(7):747–755, 2011, SIAM/ACM Joint Conference on Geometric and Physical Modeling.

[354] R. Yusuf, J. Owusu, H. Wang, K. Qin, Z. Lawal, and Y. Dong. VQA and visual reasoning: An overview of recent datasets, methods and challenges. arXiv preprint arXiv:2212.13296, 2022.

[355] S. Yuval-Greenberg, E. P. Merriam, and D. J. Heeger. Spontaneous microsaccades reflect shifts in covert attention. *Journal of Neuroscience*, 34(41): 13693–13700, 2014.

[356] B. Zhu and W. Chen. Performance histogram curve: Abstractive visualization and analysis of NBA games. *International Journal of Software & Informatics*, 10(3):11, 2016.

[357] Y. Zhu, J. Yu, and J. Wu. Chro-ring: A time-oriented visual approach to represent writer's history. *The Visual Computer*, 32(9):1133–1149, 2016.