Visual Analytics Guided Exploration and Assessment of Time Series Pattern Search Algorithms

Mick Sneekes
6321860

Daily Supervisor: Y. Yu, MSc
First Supervisor: Prof. Dr. M. Behrisch
Second Supervisor: Dr. A. Chatzimparmpas

School of Natural Sciences
Utrecht University
The Netherlands
January 31, 2025

Abstract

Many research domains deal with large amounts of chronological observations. These chronological observations are often collected into time series. An important part of analyzing time series data is performing pattern search. Provided with an example pattern, an algorithm outputs positions, sizes and confidences of similar behavior in the data. Currently, these algorithms are evaluated using numerical metrics, which can only assess the quality of the retrieved pattern and the pattern search algorithm as a whole. Numerical evaluation metrics do not provide any explanation for systematic errors made by time series pattern search algorithms. This study investigated visualization of the found similar patterns and their corresponding predetermined ground truths, including especially their positions, sizes and confidence, resulting in a visual analytics approach used to evaluate the quality of the found similarities. The proposed visual analytics approach provides detailed information about both the quality of the result and how the result is imprecise. The visualized results allow for a detailed analysis of the way a pattern search algorithm provides results, why these results are of a certain quality and where the pattern search algorithms make systematic errors in generating similar patterns. In turn, this allows for analysis of whether a pattern search algorithm works well in certain situations in combination with explainable errors, resulting in more accurate pattern search algorithms in the future. The proposed approach was evaluated in a case study and an expert study. Additionally, the proposed approach was compared to similar visualizations previously available, noting the added value of this approach.

Acknowledgements

Words cannot express my gratitude towards my supervisors for their guidance and expertise during this journey. I could also not have completed this journey without IAV GmbH, who provided knowledge and information.

Lastly, I would like to recognize the unwavering support of my family and friends, especially my parents. Their belief in me has allowed me to keep motivated during this journey.

Contents

Chapter 1

Introduction

1.1 Background

The proliferation of technology in most parts of human life has led to a large increase in the collection of data in recent history [Xu et al., 2014]. Almost every domain makes use of sequences of observations over time to better understand data within their field. These observations over time are recorded in time series. A time series is a collection of sequential data points, collected or recorded at consecutive points in time. Examples of fields that frequently use time series data are the automotive domain [Yu et al., 2023], the stock market [Fu et al., 2007] and scientific observations, often made using sensor data [Faloutsos et al., 1994]. The inherent sequential nature of time series data makes it very suitable for the analysis of trends and patterns within data. This sequential nature also introduces a specialized manner of analysis, requiring specialized methods to process, analyze and interpret time series data.

Pattern search is the problem of searching for the most similar sequences within a database of sequences, based on a given query sequence. Finding matching patterns within data is an essential part of analyzing time series data. These matching patterns can take many different forms within many different fields. Examples include trading patterns in stock market data, providing a more detailed insight into market dynamics, or measurements of body temperature and heart rate indicating certain issues within a patient. When databases get larger, being able to automatically find subsequences of data that closely match a query becomes paramount, as researchers are not able to efficiently identify patterns by hand. Being able to effectively and accurately identify matching patterns within time series data is crucial for

extracting meaningful information from these datasets.

Noise and irregularities within data can obfuscate patterns of interest within a time series. Additionally, the large size of many time series databases can prove challenging when searching for matching patterns. A range of different algorithms and models have been developed to tackle these challenges[Fu, 2011], all performing differently on different sets of data. To make further developments on these algorithms or compare different algorithms for different sets of data, specific algorithms need to be evaluated. The improvement of time series pattern search algorithms often results in different sets of potentially similar patterns. These different sets of potentially similar patterns can be the result of different versions of an algorithm being developed or query enhancement through relevance feedback, a technique often used in this field to improve the query provided to an algorithm. To evaluate or compare algorithms researchers stick to numerical metrics, often using only well known metrics like precision and recall.

While significant advancements in the field of time series pattern searching have resulted in a surge in the number of algorithms that have been developed to perform time series pattern search, tools to visualize their results are largely unexplored. Visualization plays a large role in making data more accessible.

> Visualization of data makes it possible for researchers, analysts, engineers, and the lay audience to obtain insight into these data in an efficient and effective way, thanks to the unique capabilities of the human visual system, which enables us to detect interesting features and patterns in short time.[van Wijk, 2005]

Being able to find interesting features and patterns in the results of time series pattern analysis is paramount when evaluating the quality of these results. This research aims to address the gaps left by existing evaluation methods for time series pattern search algorithms.

1.2 Problem Statement & Research Questions

The aim of this research is to design an information visualization approach, which visualizes the results of time series pattern search algorithms, along with the implementation of a tool using that proposed approach.

Due to the scope of this project being sufficiently narrow, a single research question will cover the entire research:

RQ1: Can an information visualization tool be implemented to help evaluate the results of time series pattern search algorithms?

The following four sub-questions back the main question by dividing the scope into smaller parts:

SQ1: Which tools for evaluating results of time series pattern search exist and what are the gaps between the existing tools and the practical application need?

SQ2: What specific tasks must a visual evaluation tool achieve to allow users to evaluate time series pattern search algorithms?

SQ3: What are effective visualization techniques for evaluating the results of time series pattern search algorithms?

SQ4: Does the proposed visual analytics approach provide additional insights into the results of time series pattern search algorithms in real-world case studies?

1.3 Goals and Scope

To determine relevant goals for the visualization approach, the MoSCoW prioritization method was used. The MoSCoW prioritization method allows for strict focus on the most important parts of the approach, while still allowing room for additional aspirations. The first subsection below details the Must-have, Should-have and Could-have goals for the project. Within these larger categories, the goals are grouped by their stage within the visualization design process. The specific stage each goal belongs to can be found at the end of the goal, inside square brackets. These stages were based on the Nested Model, proposed by Munzner [2009], adapted to better fit the specific list of goals for this research. Besides *Evaluation* and *System Design*, the stages were directly derived from the Nested Model. The *System Design* stage is the stage in which design choices are made regarding the carrier of the visualization system and what specific visualizations it will contain. In the *Visualization Design* stage, specific visualization elements are chosen or designed to represent specific elements in the data. The *Interaction Design* stage contains the choices made for the interactions that are available for the designed visual elements and the interactions between different visualizations. Only a minor part of this research, the *Algorithm Design* stage concerns itself with the design of specific algorithms required to make the visualizations work. The *Evaluation* stage concerns itself with the evaluation

of the visualizations and the visualization system. Finally, the subsection scope contains more information on the Won't-have part of the MoSCoW prioritization method. This will provide more detail on the boundaries of the approach.

1.3.1 Goals

Must have Goals

MH1: The master student must design and implement an application as the carrier of the visual analytics approach. [System Design]

MH2: The designed application must be able to visualize the time series, the query (the pattern to search for), the ground truths (predefined expected found patterns), and the found patterns containing predicted similar behavior, as glyphs showing their essential features (like the lengths of the found patterns and the ground truths as well as the relative positions between matching pairs of found patterns and ground truths) in a chart (hereinafter called time glyph chart, as the patterns are arranged based on their occurring time). [Visualization Design]

MH3: The application must contain functionality for visualizing a found pattern with its matching ground truth. [Visualization Design]

MH4: The application must be able to visualize the found patterns as aligned glyphs (hereinafter called aligned chart) because users may not be interested in the temporal positions/order of the patterns. [Visualization Design]

MH5: The aligned chart must be able to visualize the confidence of the found patterns. [Visualization Design]

MH6: The aligned chart must be able to visualize the ground truth - found pattern pairs (potential true positives, correctly found patterns), found patterns without matching ground truths (false positives, incorrectly found patterns), and ground truths without matching found patterns (false negatives, missing patterns). [Visualization Design]

MH7: The glyphs / found patterns in the aligned chart must support sorting according to useful criteria, e.g., time, pattern lengths, and confidence. [Interaction Design]

MH8: The glyphs in the aligned bar chart must, upon hovering over or clicking on, support showing a line chart visualizing the original pattern with a bit of context before and after the start of the pattern so that if the glyph chart guides the analyst to an interesting pattern/phenomenon, the analyst can examine the details. [Interaction Design]

MH9: The application must be able to match found patterns with their corresponding ground truths algorithmically, as a preparation for analysis of false positives, false negatives, and general alignment analysis. [Algorithm Design]

MH10: The master student must conduct a case study showcasing the features of the app. [Evaluation]

Should have Goals

SH1: The master student should conduct a literature review on the current evaluation methods/metrics for time series pattern search, e.g., average precision and top-5 recalls with ground truths, deviation of found patterns compared with the baseline (in the case the proposed method is an accelerated version of the baseline method with potential accuracy loss), and visual inspection. [System Design]

SH2: The master student should augment the aligned chart for method/-model comparison. [Visualization Design]

SH3: The master student should augment the aligned chart for results from multiple relevance feedback rounds. [Visualization Design]

SH4: The master student should implement a line chart showing the selected time series data. The line chart should show novel visual encodings for ground truths and found patterns (un-)matching. [Visualization Design]

SH5: The different panels, views and charts should have a meaningful and aesthetically pleasing layout. [Visualization Design]

SH6: The application should visualize a confidence threshold in the aligned chart sorted according to confidence (with the overlaid confidence line chart), and support tuning the confidence threshold and the number of found patterns. Changing to a higher confidence threshold shows fewer patterns, reducing false positives but introducing potential false negatives. [Interaction Design]

SH7: All charts including the line chart and the glyph charts should communicate with each other, e.g., when clicking a glyph in the aligned chart, the corresponding pattern will be zoomed in on and highlighted in the line chart; conversely, when clicking a ground truth / found pattern in the line chart, the corresponding glyph in the aligned chart should be highlighted. [Interaction Design]

SH8: The master student should extend the case study to the new visualizations. [Evaluation]

SH9: The master student should design a user study (with components like predefined tasks, predefined questionnaires, think-aloud, etc.) and conduct the study. [Evaluation]

SH10: The master student should conduct a design space review on similar existing visual designs, like LineUp. [Evaluation]

Could have goals

CH1: The master student could take it to the next level and develop a leaderboard app showing the performance of various methods on diverse datasets with multiple metrics, which not only shows the performance as numeric values but also integrates the visual analytics-guided tool for error diagnosis/analysis. [System Design]

CH2: The master student could propose alternative designs for important visualizations and interactions with arguments for and against them. [Visualization Design]

CH3: More features could be visually encoded or encoded in an alternative way. [Visualization Design]

CH4: The approach allows us to understand the evolution of relevance feedback for the given case. [Evaluation]

1.3.2 Scope

This research focuses solely on the visualization of results of time series pattern search algorithms to evaluate these algorithms. As a visualization approach, it will not contain any tools for or research into doing time series pattern search, unless necessary for the design or implementation of the visualization approach. This research only uses very minor knowledge of time

series pattern search algorithms to show their results, which are provided to the tool before operation. The implemented tool will not be able to do time series pattern search.

Existing research often evaluates time series pattern search algorithms on their performance. Very large datasets are used when performing pattern search, which means the efficiency and speed of the algorithm are very important factors when comparing algorithms. Since this project will not perform time series pattern search itself, only provide a way of evaluation algorithms that perform time series pattern search, response time, memory efficiency and overall speed are not used as evaluation metrics for these pattern search algorithms.

1.4 Contribution Statement

This research provides contributions to the field of time series analysis by addressing the challenge of visualizing the results of time series pattern search algorithms. The most important contributions of this research are as follows:

- Provide insight into the shortcomings of state-of-the-art evaluation methods for time series pattern search.
- This research will provide novel interpretations of the results of time series pattern search algorithms.
- By providing a novel approach and tool for evaluating time series pattern search algorithms, researchers will be able to more easily and accurately improve their algorithms.
- By allowing new users to visually evaluate the results of a time series pattern search algorithm, it lowers the barrier for new users to choose the algorithm that best fits their needs.
- This research will provide a basis for future work on the evaluation of time series pattern search algorithms, focused on the visualization of results.
- Domain-specific case studies are presented, which showcase the impact of the visualization approach and tool in real-world scenarios.
- An expert study showing an example of a potential end-user working with the implemented approach.

1.5 Thesis Outline

First, Chapter 2 will contain a brief literature review. This literature review will analyze the state of the art in evaluating time series pattern search algorithms and visualizing their results.

Afterward, a task analysis will be performed on the domain of time series pattern search result visualization and evaluation in Chapter 3, using the goals as detailed below in Section 1.3.1.

Next, Chapter 4 will provide a detailed explanation of the design and implementation choices involved in the creation of the visualization approach and its implementation.

Chapter 5 will contain multiple case studies, evaluating the proposed implementation and visualization approach in real-world scenarios.

Lastly, Chapter 6 will contain a discussion on current limitations and possibilities for future work, ending with some concluding words.

Chapter 2

Related Work

This chapter contains a literature study, detailing different aspects of time series pattern search algorithm evaluation, time series visualization and other research which touches on the visualization of time series pattern search results. The first section contains a detailed analysis of the state of the art in the evaluation and visualization of time series data, followed by a list of gaps to be improved upon.

2.1 State of the Art

To evaluate the different time series pattern search algorithms, different metrics are used. During the literature review, no overview papers of evaluation methods or papers comparing multiple algorithms were discovered. Due to this limitation, the state of the art in evaluating time series pattern search algorithms was explored using technique papers proposing algorithms for time series pattern search. When proposing novel algorithms for time series pattern search, often a section is dedicated to the evaluation of the proposed method or the comparison of the proposed method to the state of the art. Firstly, an overview is made of different metrics used in different papers, providing a list of currently used metrics for evaluating time series pattern search algorithms, followed by an overview of efforts in the visualization of time series data.

2.1.1 Algorithm Evaluation

Often, identifying a pattern as similar or matching is deemed the most important goal of time series pattern search algorithms. To evaluate classification

algorithms, researchers often use a confusion matrix. A confusion matrix compares the patterns found by an algorithm to the predetermined ground truths, noting whether a found pattern should be found according to these ground truths. Figure 2.1 shows a confusion matrix tailored to time series pattern search. As can be seen in the figure, when a found pattern matches a ground truth, it is called a True Positive (TP), while a found pattern that has no matching ground truth is called a False Positive (FP). Furthermore, a ground truth that has no matching found pattern is called a False Negative (FN), while every subsequence that is not a ground truth and doesn't have a found pattern is called a True Negative (TN). Since time series pattern search algorithms do not discern True Negatives, these are not used in evaluating time series pattern search algorithms.

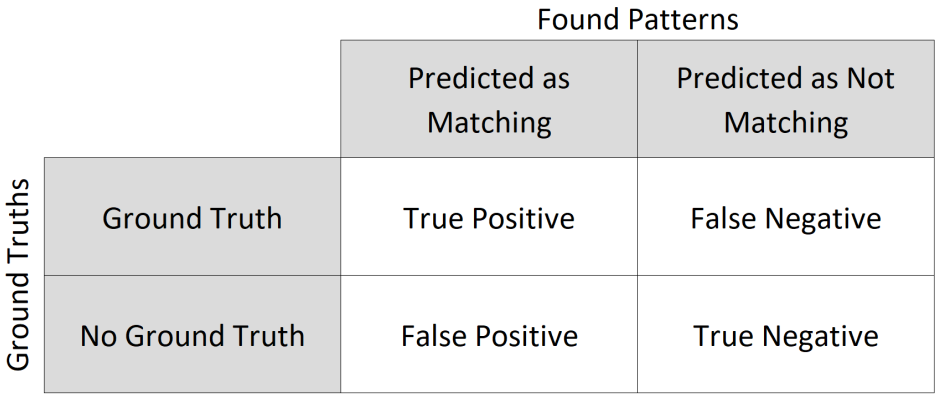


Figure 2.1: Confusion matrix for time series pattern search.

Derived from the confusion matrix, the metrics Accuracy, Precision and Recall are often used in evaluating time series pattern search algorithms. How to calculate each of these metrics can be found in Equation 2.1, Equation 2.2 and Equation 2.3. As explained before, these metrics only care about whether a ground truth has been found, not in what manner or how much of the ground truth. While this can be fitting for some types of research, more information on the results of the algorithm are often required. Examples of these metrics being used can be found in Kahveci and Singh [2004], Fu et al. [2007], Papapetrou et al. [2011], Wan et al. [2016], and Janka et al. [2019] but many more examples of the use of these metrics are available.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

Gong et al. [2016] use both Accuracy-on-Retrieval (AoR) and Accuracy-on-Detection (AoD). The AoR measures how well the algorithm can retrieve patterns that are expected to be retrieved. AoD measures the quality of the retrieved patterns. In essence, the AoR behaves similarly to recall, comparing the set of retrieved patterns to the set of patterns that are expected to be retrieved, adding a threshold value that defines when an expected pattern is retrieved. The AoD divides the sum of each retrieved pattern's overlap percentage by the amount of patterns retrieved.

Yu et al. [2023] adds the use of mean Average Precision (mAP) to the pattern search space. mAP is frequently used to evaluate machine learning models, calculating the average precision for different classes and using these values to calculate the mean Average Precision for a model. This essentially allows for simple scoring of algorithms, resulting in easy comparison of different machine learning models. While mAP was a more accepted evaluation metric for object detection models in computer vision, it was rarely used outside of this. Additionally, Yu et al. use balanced accuracy, a combination of the true negative rate and recall, accuracy, precision, recall and F1-score. The F1-score is the harmonic mean of an algorithm's precision and recall, explained in Equation 2.4.

$$F_1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.4}$$

In 2020, Lekschas et al. [2020] proposed a method for binary classification which was to be used to find regions of time series that exhibit similar behavior as a query region. To evaluate their proposed method, compared to existing time series pattern search algorithms, a user study was conducted. 75 participants were shown a query pattern together with its five most similar patterns according to different similarity comparison models. These participants were asked to identify the group of patterns that on average most resembled the query pattern, based on a visual comparison. When the original evaluation showed unexpected results, a new user study was conducted which only compared a single similar pattern to a query. Additionally, the method was evaluated during an expert study, which asked experts to perform two tasks using the proposed method. To compare simulated data to data generated by experts in this study, they used the area under the receiver operating

characteristics curve (AUC-ROC), average precision and Matthews correlation coefficient (MCC). The AUC-ROC plots the recall against the False Positive Rate, calculated by dividing the number of False Positives by the total number of positive cases. By calculating the area under this curve, a score between 0 and 1 is calculated. The MCC is the closest one can get to a single number representing an entire confusion matrix. Using all parts of the confusion matrix in a single calculation, the MCC is regarded as one of the most informative single score values to evaluate binary classification algorithms.

Mannino and Abouzied [2018] presented Qetch, a tool that used user-sketched patterns as the query for time series pattern search. Qetch used a novel matching algorithm that takes human error in drawing into account when searching for similar patterns in time series data. To compare the proposed matching algorithm to existing methods, Mannino and Abouzied defined a Precision@5. Matching algorithms were tasked with searching for a reference pattern in a time series, being provided a sketch as a query. The Precision@5 value ranges from 0 to 1, with higher values showing that a matching algorithm more often found the reference pattern in its top 5 results, given different sketches.

Many papers evaluate time series pattern search algorithms in terms of efficiency or speed. Due to the efficiency and speed of algorithms being out of the scope of this research, these values will not be considered further.

2.1.2 Visual Query Systems for Time Series Data

Visualizing results of time series pattern search algorithms is largely an unexplored field. A Visual Query System (VQS) is a tool, built to aid data querying with a visual representation of the data and the query, as compared to text-based query languages. Using a VQS for performing time series pattern search is a field more explored compared to result visualization. While VQSs are not designed for the same use case as visual evaluation systems, they often visualize a subset of the results of pattern search. This similarity makes them an appealing alternative for comparison against a proposed visual evaluation system. In this section, a list of time series VQSs created for time series pattern search is provided, detailing their stand-out features, mainly concerning the visualization of the results of time series pattern search.

Drawn Queries

One of the ways a query pattern in time series data can be visually defined by users is by drawing or sketching. SENSOR, as proposed by Yan et al. [2022], is one of these VQSs using user drawing to define a query for time series pattern search. SENSOR visualizes time series datasets as a line graph. This line graph allows users to interact with and explore the dataset, providing zooming and panning as interactivity options. Below this line graph, a canvas is provided, allowing users to freehand sketch a desired query for doing time series pattern searching. Next to these main two visualizations, SENSOR displays a list of common shapes, defined by Gregory and Shneiderman [2012], and shapelets selected from the data. SENSOR allows users to drag-and-drop these common shapes or shapelets to the canvas, using the common pattern or shapelet as the query for time series pattern search. SENSOR does not include the capability of visualizing the results of time series pattern search or the visualization of predefined ground truths, being designed as an interface for query formulation and not an interface for the visualization of results.

Qetch, proposed by Mannino and Abouzied [2018], allows for the definition of queries in a similar way. At first, Qetch visualizes the data in the dataset in a line graph, along with a canvas for providing the system with a query. Using user-drawn sketches, slightly smoothed to correct hand jitter, as query, Qetch provides matches both in the time series dataset and a library of frequently used queries, saved queries and mathematical functions. After fine-tuning the query, possibly by selecting one of the predefined patterns in the query library, Qetch searches for similar patterns in the dataset. The results of this time series pattern search are provided in a table and visually in the time series line graph. Using the color of the visualized patterns to display the goodness of the match, matching patterns are overlayed over the time series data in the line graph. While Qetch uses intuitive visualization techniques to show the results of its included pattern-matching algorithm, it lacks the capability of visualizing the relative position of predefined ground truths.

In their paper researching ambiguity in sketching queries for time series pattern search, Correll and Gleicher [2016] presented a web-deployed prototype for a sketch-based VQS. They used a visualization technique called small multiples for the visualization of patterns in a time series dataset. By using multiple smaller visualizations, all using the same scale and axes, they can create a scrollable visualization, visualizing even the lowest quality matches found by time series pattern search. These patterns are color-coded

according to their quality. Additionally, they use a separate visualization to only view the top results, allowing users to select the specific amount they want to see. The last visualization is again a drawing canvas, providing no way of visualizing predefined ground truths.

QuerySketch was one of the first tools to use a hand-drawn sketch as the query for searching historical stock price data. Proposed by Wattenberg [2001], QuerySketch generates a list of thumbnail-style graphs, giving users a small representation of a similar pattern to their query. By hovering over one of these thumbnails, a larger representation of the time series subsequence is visualized over the drawing area. Lastly, clicking one of the thumbnails provides users more context by referring to information on a stock research website. QuerySketch provided a novel way to visualize similar patterns by using thumbnails, providing details on demand when hovering, but it did not include visualizations for entire time series or predefined ground truths.

Eichmann and Zgraggen [2015] used a visualization tool to evaluate different similarity measures for time series pattern search, called TimeSketch. This tool allowed users to query a time series using hand-drawn queries, afterward displaying a ranking of similarity measures. The ranking was found insufficient at the point of writing. TimeSketch does not provide a visualization for the entire time series data, removing this context from the found patterns, neither does it contain functionality for displaying predefined ground truths.

Timeboxes

Besides using human-drawn sketches as queries, another possibility for visual query definition is using direct-manipulation rectangular Timeboxes. Timeboxes were defined by Hochheiser and Shneiderman [2001]. In simple terms, a Timebox introduces constraints on the time series, used to generate a query. The width of a time box defines the range of time in which is to be searched, while the height of the box defines the actual values that are to be searched for in that timeframe. Combining multiple of these Timeboxes can create queries of increased complexity, while still being easier to interpret compared to their mathematical representations. These Timeboxes were used in TimeSearcher [Hochheiser and Shneiderman, 2002] and TimeSearcher2 [Buono et al., 2005] to allow users to graphically create a query for time series pattern search. The original TimeSearcher used a set of user-defined Timeboxes to identify similarity in entire time series. It uses two visualizations to show the data found within multiple time series datasets or different variables in the same dataset, one for an overview of all time

series and one for drawing Timeboxes. The first visualization updates when a query is executed, showing the matching datasets instead of every data set. TimeSearcher2 introduced the ability to search for similar patterns within time series data. Again, users can draw Timeboxes over a line graph to be used as a query. After executing the desired query, an overview visualization shows all locations of similar patterns in the data, allowing users to view details of those patterns in the line graph by moving a field-of-view box.

Removing the height of Timeboxes, PSUEDo, proposed by Yu et al. [2022], uses a subsequence of the entire time series to define a query. Users can configure a line graph view of the time series, using a minimap and numerical parameters to define a subsequence of the time series to be displayed. This subsequence can be used as a query for time series pattern search. PSEUDo additionally visualizes the query sequence in a separate view, removed from the rest of the time series, along with both a visualization showing filtered results and a visualization showing all results of the pattern search algorithm. The filtered results provide users with color-coded samples of results of the pattern search algorithm, showing their relative similarity through color. While PSEUDo uses compelling visualizations for the results of time series pattern search algorithms and their quality, it does not include any way to visualize ground truths along with these results.

2.1.3 Other Time Series Visualization Tools

Tools exist that visualize time series, which are not VQSs. Two of these tools and their comparison to a visual evaluation tool will be explained below.

Hao et al. [2011] propose a visualization tool, that analyses time series data and provides a visual representation of frequently found patterns, often called motifs, in the data. Compared to pattern search, motif detection does not rely on a query to search for patterns in time series. Instead, it looks for patterns that occur frequently in the time series. While this tool does not provide any way of visualizing the results of time series pattern search algorithms, the visualization they created does face similar challenges and uses similar techniques as a potential result visualization could use. Using different colored rectangles, combined with the merging of overlapping motifs the tool provides an intuitive view of where frequent patterns are found within a dataset, even allowing users to change several parameters to provide them the best view.

LiveRAC, proposed by McLachlan et al. [2008], is an analysis tool created to analyze large collections of system management time series data. LiveRAC used semantic zooming, changing visualizations according to the

space provided to them, to create a high-density matrix of time series data. Not having anything to do with time series pattern search, LiveRAC does not visualize the results of time series pattern search algorithms or predefined ground truths. It does, however, use engaging interactions and filtering methods to create a very high-density visualization of multi-variate time series data.

2.2 Gaps

To be able to create a list of tasks for a proposed visualization approach, as well as evaluate the approach in the end, the gaps in the state of the art are required to be listed. In this section, the most important gaps that exist in current evaluation techniques will be listed and explained, relating to the creation of a visual analytics approach.

The most important gap in the state of the art is the lack of localized evaluation techniques for time series pattern search algorithms. Every metric listed in Section 2.1 evaluates an algorithm as a whole, while disregarding local errors in finding similar patterns. For these global metrics, determining if a specific found pattern matches a ground truth is enough to calculate a score for the entire algorithm. No consideration is given to how the found pattern matches the ground truth. Depending on the way a found pattern is determined to match a ground truth, a lot more information is hidden by just branding it a match and not looking at how the found pattern matches. For example, if a found pattern matches a ground truth when it covers the entire ground truth, no consideration is given to any excess data found besides the ground truth. Similarly, a found pattern may only find part of a ground truth, and thus be deemed non-matching. This information is lost in the numerical metrics considered in the state of the art.

Even when looking at the actual results of a time series pattern search algorithm, the numbers listed will make finding patterns very difficult. While every bit of information is covered by the raw results provided by an algorithm, it is very hard to see where errors lie and if these errors are systematic or irregular. To achieve the easiest and most intuitive analysis of the results, a visualization tool would be required.

Regarding the visualization of time series pattern search results, no system was found that allowed for the visual comparison of found patterns and expected patterns. Overall, visual analysis of time series pattern search results is a largely underexplored field that provides plenty of opportunities for innovation. While visual evaluation of the retrieved results is done in some

way by existing visual exploration tools, no tools exist that match retrieved patterns with their corresponding ground truths, enabling precise localized evaluation of the results of time series pattern search algorithms.

To summarize, the state-of-the-art lacks localized evaluation metrics for time series pattern search algorithms. Additionally, it would benefit largely from the more intuitive pattern analysis provided by visualized results to find systematic errors.

Chapter 3

Task Analysis

Doing proper task analysis is an essential part of creating an effective and user-centric design for the visualization approach. Task analysis provides a semi-structured overview of suspected user goals within a system, allowing for insight in how the tasks fit together. This chapter aims to analyze the tasks involved in visually evaluating the results of time series pattern search algorithms. The task analysis will provide critical insights into requirements and challenges in the visual evaluation of time series pattern search algorithms, allowing the task analysis to be used as guidelines for the design and evaluation of a visual analytics approach. To define a list of tasks, the list of goals, described in Section 1.3.1, was analyzed. To refine and validate the list of tasks, the goals were discussed with a domain expert with experience in the field of designing and evaluating time series pattern search algorithms. This ensured that the goals were realistic and in line with domain-specific requirements. Domain experts were regarded as the target users for the task analysis, which allows the assumption that users have a high level of experience within the domain. This eliminates the need for the highest granularity tasks, by assuming user familiarity with similar tasks. The list of tasks is in no way comprehensive and largely biased by the domain of this research.

The chapter will begin with a list of tasks and sub-tasks with a brief explanation, followed by a description of areas for future expansion.

3.1 Tasks

This section contains a list of tasks for the evaluation of the results of time series pattern search algorithms. The ordering of the tasks is mostly done in a sequential order, meaning later tasks depend on the completion of earlier

tasks. Additionally, this allows for the iteration of tasks when necessary, being able to loop back to earlier tasks without losing the consistency of the sequence.

The list is further divided into sub-tasks where possible. These sub-tasks are optional parts of the main task unless stated otherwise. The sub-tasks are not listed in the same sequential manner as the main tasks, meaning the order of the sub-tasks is not important.

Where possible, tasks will be related to corresponding goals. Each task will explain its link to a specific goal, using the identifier found in Section 1.3.1. Further explanation of each goal can also be found in this section.

T1: Data validation and initial exploration

The initial analysis of the visualizations allows users to verify that the uploaded data is the correct data, while also providing first insights into the results of the time series pattern search algorithm. Without having to adjust any settings in the visualization approach, users will be able to view the visualizations, using the gathered information to do some initial analysis of the used algorithm.

T1.1: Validate the uploaded data

T1.2: Draw initial conclusions for visualized data

T2: Interact with data

To gain more insight into the workings and results of the time series pattern search algorithm, the visualizations can be adjusted to fit the needs of the researcher. Different visualizations will need to interact with each other at the same time as being interacted with by the researcher. Researchers can also personalize some parts of the approach, to better fit the goals of their evaluation. Besides changing the way data is shown, researchers can also filter the data shown in the visualizations. Not all visualized information will be important to researchers. To allow for better evaluation, the data can be filtered to only include the data that the researcher finds important for their evaluation. The main goals included in this task are **MH7**, **MH8**, **SH4**, **SH6**, **SH7**, **CH2** and **CH3**. All these goals refer to different ways of filtering, sorting or visualizing data. Manipulating how the data is visualized allows for the dynamic testing of hypotheses and assumptions, along with providing instant feedback for further decision-making.

T2.1: Change sorting of ground truths and found patterns

T2.2: Change zoom level of time series

T2.3: Change which variables are visualized within the time series

T2.4: Personalize visualization techniques

T2.5: Filter only high or low confidence found patterns

T2.6: Filter only high or low quality found patterns

T3: Evaluate results for individual algorithms

When all visualizations are adjusted and all filtering is done, researchers want to find patterns in the results of the algorithm that is being evaluated. This means being able to view the specific location of found patterns and ground truths in the time series, both for single instances or on an overview level, depending on choices made while adjusting the visualizations. Being able to view all true positives, false positives and false negatives visually allows for the further evaluation of more global metrics, for example existing numerical metrics like precision and recall. Goals **MH6**, **MH9**, **SH4** and **SH7** resulted in this task. All of these goals refer to the visualization of true positives, false negatives, false positives and the overview of the time series. Eliciting meaningful patterns from numerical results is very difficult. Using visualized representations of the results makes eliciting meaningful patterns more intuitive and easy to achieve.

T3.1: Identify patterns in true positives

T3.2: Identify patterns in false negatives

T3.3: Identify patterns in false positives

T3.4: Cross-reference abstract pattern data to specific time series data

T4: Qualitatively compare results for different algorithms

Researchers may want to compare the results of different algorithms, for example to identify the most appropriate one for their research. To compare the results of different algorithms, the results gathered in task **T3** could be compared to the results gathered for different algorithms. This includes both visualized results and numerical metrics, calculated using the visualized results. This task is a direct result of goal **SH2**.

T4.1: Compare visualized results of different algorithms

T4.2: Compare global metrics for different algorithms

T5: Qualitatively compare results for different versions of an algorithm

To further improve the results of a single algorithms, different version need to be tested and compared. Similarly to comparing different algorithms, the results gathered in the **T3** task could be compared for different versions of the same algorithm. Comparison of both the visualized results and global metrics, calculated using the visualized results, can be used to compare different versions of the same algorithm. This task is a direct result of goal **SH3**.

T5.1: Compare visualized results of different algorithms

T5.2: Compare global metrics for different algorithms

3.2 Future expansion

In this section, some of the tasks that were considered to be a possible task within a time series pattern search visualization approach, but were not included in this research.

The first task that was considered but not included is *Exporting insights*. To more accurately keep track of the results of evaluating time series pattern search algorithms, being able to export the results of the visualization may provide a way to maintain a database of different evaluations for, for example, different algorithms. While users are able to use screenshot functionality, built into their operating system or otherwise, to export and save visualizations, the task of exporting and importing previous visualization results is not considered in the design of the proposed approach.

Secondly, the task of *Backtracking* was not considered for this research. Backtracking is the act of retracing one's steps, making users able to view previous adjustments of visualizations. Being able to retrace the steps taken while adjusting visualizations can undo adjustments that are deemed as not helpful, bringing users back to a set of adjustments that they were more comfortable with or that better fit their needs.

Chapter 4

Visualization of Algorithm Quality Exploration and Comparison

This chapter will explain the design choices made in developing the visualization approach, along with details on algorithms and visualizations that were developed while implementing the approach. In planning, the tool was to be divided into three different modules, each serving a relevant use case, gathered from the goals. During design, the choice was made to merge the last two modules. This choice will be explained further in Section 4.4. All modules were developed to handle a different subset of tasks. A detailed explanation of these tasks can be found in Chapter 3. While some tasks are covered in multiple modules, some can only be covered by a single module. First, an overview of these modules will be provided, explaining which visualizations are developed for the visual system and what their major functions are. The different visualizations will be grouped by the tasks considered in their design. Afterward, a more detailed explanation of design choices within each visualization will be provided.

The visualization approach was implemented using a combination of HTML, CSS and JavaScript. To create the visualizations in JavaScript, version 7.9.0 of the D3.js visualization library was used. In addition to custom CSS, the Bulma CSS framework was used to create a pleasing layout and styling for HTML elements within the implementation.

To generate screenshots of the final implementation the EEG Eye State [Roesler, 2013] dataset and the Deep Valve dataset were used, which will be explained in more detail in Section 5.1. Algorithms important to the

functionality of the approach will be explained using pseudo-code versions of these algorithms. The pseudo-code algorithms are loosely based on JavaScript algorithms.

4.1 Overview

To better provide an overview of visualizations, this section is grouped by the different modules in the system. The first module, called the *Algorithm Exploration* module from here, is concerned with the exploration of the results of a single time series pattern search algorithm. The second module is made to compare the results of different time series pattern search algorithms, which is the reason it is named the *Algorithm Comparison* module in the future. The last module is called the *Version Comparison* module. This module was planned to be designed to provide a comparison between different versions of the same algorithm, in particular being improved by using relevance feedback, as can be found in goal **SH3**. The last module is closely related to the second, both visualizing different sets of results. Each module was designed while keeping in mind a subset of the tasks defined in Chapter 3. Which task is covered by which module can be seen in Table 4.1.

	T1	**T2**	**T3**	**T4**	**T5**
Algorithm Exploration Module	x	x	x		
Algorithm Comparison Module	x	x		x	
Algorithm Version Comparison Module	x	x			x

Table 4.1: Module task coverage

Primarily, every module of the visualization approach uses the same visualizations as building blocks, behaving differently within different modules. The main visualization within each module is the glyph chart, created to illustrate the quality of the retrieved results, without including overwhelming unnecessary noise. This glyph chart uses different visual encodings to show the relative position of one or more found patterns and their matching ground truths. Goals **MH2** through **MH7** were the main considerations when creating the glyph chart.

To further contextualize the glyph chart, both modules use a line chart and a histogram. The line chart visualizes the time series itself, with the addition of the found patterns and ground truth. Goals **MH8** and **SH4** were the main considerations when creating this line chart. The histogram

visualizes the distribution of the confidence for all found patterns. Goals **SH6** and **CH2** were the inspiration for the design of this histogram.

4.2 Algorithm Exploration Module

The first module that will be discussed is the *Algorithm Exploration* module. This module was designed to provide a view of the results of a single algorithm, providing users the ability to explore these results. A screenshot providing an overview of the implementation of the *Algorithm Exploration* module can be seen in Figure 4.1. Different components of the overview are labeled with letters, which will be used to explain them further in this section.

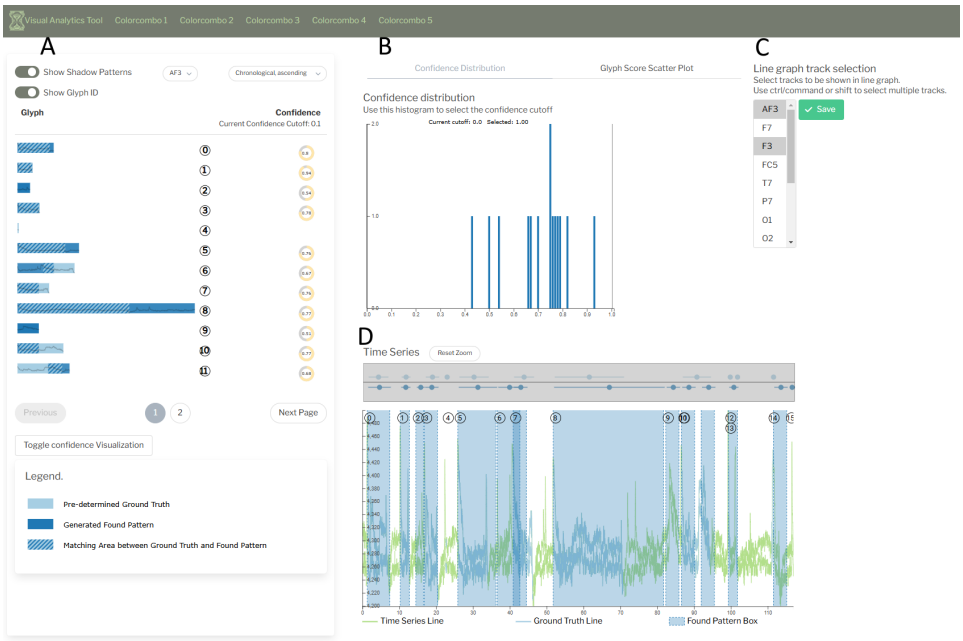


Figure 4.1: Overview of the Algorithm Exploration Module. A larger version of this figure is available in Appendix A.

Module Overview As discussed in Section 4.1, the implemented tool contains all major visualizations. The glyph chart, labeled A in Figure 4.1, shows the relative position of all found patterns and ground truths, matched together in a single visual symbol where possible. Labels B and D show

the confidence histogram and the line graph. In addition to these major elements, two ways of personalizing the visualizations are found in this module. Firstly, users can choose which variables are shown within the time graph by using the *Line graph track selection*, labeled by the letter C. This track selection element lists all variables found within the input data, allowing users to make a custom selection of the variables most important to them at that point. Secondly, users are able to switch the confidence histogram to a scatter plot, allowing for a different way of filtering data. This scatter plot generates a score for each pair of matching found patterns and ground truths, allowing users to only look at a custom selection within the glyph chart. A more detailed explanation of each major visualization in the module, along with an explanation of how they function, can be found below.

4.2.1 Glyph Chart

The primary visualization in the exploration tool is the glyph chart, found in Figure 4.1.A. The glyph chart is filled with visual symbols, which visualize a combination of ground truths and found patterns. These visual symbols take the shape of colored rectangles, which show either a found pattern or a ground truth or both. Each row contains a combination of one or more rectangles, showing where a found pattern matches a ground truth. One such combination will be referred to as a glyph from here. When starting, rows encode the temporal location of these glyphs, ordered by either the start of the found pattern or the start of the ground truth, using the smallest value when sorting. Different sorting options are available, which will be explained later, changing the encoding of the rows.

Pattern Abstraction To identify patterns in the workings of time series pattern search algorithms, the value of the variable in the time series is not required. The decision was made to primarily visualize the temporal location in the time series, without the noise of the variable itself being included. Visualizing found patterns and ground truths as bars or rectangles in a table removed even the chronological aspect of the glyphs. The main consideration for these bars is their length, which visualizes the length of the found pattern or ground truth, and the relative temporal location of a matching ground truth and found pattern.

Generating Color Codings To properly differentiate between a found pattern and a ground truth within the glyphs, the choice was made to use a color encoding. Figure 4.2, below, shows the shortlist of considered color combinations. As stated by Stone [2006], even for a professional graphic

artist, choosing an existing color palette is oftentimes easier than defining one yourself. Stone recommends "ColorBrewer" by Cynthia Brewer et al. for finding good examples of color palettes. The shortlist of color combinations found in Figure 4.2 was generated using this website. ColorBrewer allows for the generation of color combinations according to some parameters set by the user. In this case, the choice was made to generate color combinations for three data classes of qualitative data. ColorBrewer uses three data classes at minimum, which exactly provide different colors for the ground truths and found patterns, in this case for the glyph chart, as well as the time series data, which will be explained further in the section on the time series graph.

Color Considerations When comparing the shortlist in Figure 4.2, not all combinations were deemed equally clear for creating glyphs. The more muted colors, as found in combinations 3, 4 and 5, lack the clarity that combinations 1 and 2 have. This effect becomes more apparent when the glyph visualization becomes more complex. Additionally, combinations 1 and 5 were deemed inappropriate due to the implications resulting from using red or orange in combination with blue and green. The specific colors in combinations 1 and 5 invite a positive-negative implication, which could distract from the intended information in the visualization. When considering the need for two colors, combinations 1 and 2 were deemed to be clear enough for this style of visualization, disregarding the implication of combination 1. Adding in a third color from the ColorBrewer palette, combination 2 was found to be the most clear for this application. While a third color is not directly necessary for the glyph chart, the need for a third color will be explained in a later section. According to ColorBrewer, when considering three colors, combination 2 is colorblind-friendly and will work well with LCD screens. This was not a main consideration when choosing the color combinations, but it is nice to have when regarding end-user inclusivity.

Figure 4.2: Comparison of combinations of colors for the glyph chart

Encoding Overlap Using colored rectangles for found patterns and ground truths introduces a new challenge when visualizing glyphs. As can be seen in Figure 4.2, the overlap between the found pattern and ground truth is no longer visible within the glyphs. Two encodings were considered to reintroduce the overlap to the glyphs. These main two options can be found in Figure 4.3.

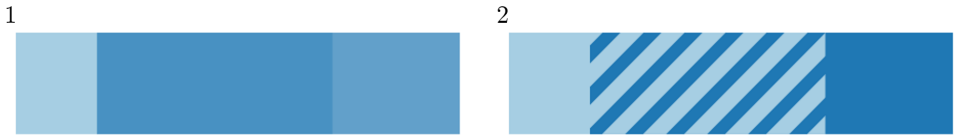


Figure 4.3: Comparison of overlap visualization options

Overlap Considerations Early prototypes used the opacity of the found pattern rectangle to show overlap between the ground truth and found pattern. An example of using opacity to visualize overlap can be seen in glyph 1 in Figure 4.3 When using very different colors for the visualization of the ground truth and found pattern, this shows very clearly where the overlap begins and where it ends. As can be seen in glyph 1, using opacity becomes less clear when using more similar colors for the ground truth and found pattern. Additionally, using the opacity in this way essentially introduces a third color for the overlap, by overlaying a less opaque color over the color for the ground truth. When compared to the crosshatch pattern used in glyph 2, it was quickly found that this provides an overlap visualization which is easier to compare between glyphs, while maintaining the precision that was achieved in glyph 1. Using a crosshatch pattern also intuitively conveys to

users that the area is part of both parts of the glyph. This is also a benefit of using a crosshatch pattern over opacity, where the third color introduced may prove more confusing to users.

Glyph Context As explained previously, the choice was made to show the found pattern as a dark blue rectangle, the ground truth as a light blue rectangle, and the overlap between a ground truth and a found pattern as a diagonal crosshatch pattern of their corresponding colors. During the development of the glyph chart, it was determined that the glyphs required a more visible link to the time series. Without this visible link, glyphs were deemed too abstract, making working with the glyph chart unintuitive. Using a visible link to the data provides the additional benefit of allowing users to quickly refer to the actual data when viewing an interesting glyph.

Figure 4.4: Example of a glyph with visible link to the data

Context Line Graph To achieve this visible link, within each glyph a line graph was added, which shows the section of the time series that is covered by the glyph. An example of a glyph with this line graph can be seen in Figure 4.4. This line can be turned on and off with a switch above the glyph chart, allowing for a more abstract view when desired. In the case the data contains multiple variables, the line in the glyph can be configured to show any of the available variables in a dropdown above the glyph chart. This allows users to isolate interesting results and compare them in one single visualization.

Confidence Visualization and Comparison

Besides the glyphs themselves, the glyph chart contains a bar showing the confidence of the found pattern within each glyph. This allows users to intuitively compare the confidence of each glyph to another glyph. Multiple implementations to show the confidence of glyphs were considered when creating this visualization. The three main candidates can be seen in Figure 4.5. The first implementation made for the tool looked like the first candidate. Being just a simple number, including the value as text in the chart may

seem sufficient. While this is a valid candidate, having the benefit of being of a low level technically, a visual component should allow users to more easily compare multiple instances of the confidence value in a quick look. When considering a visual implementation, two main components were considered, which can be seen in visualizations 1 and 2 in Figure 4.5. Screen space within the glyph chart is limited. To provide a more broad overview of the algorithm, more visualizations need to be on screen at the same time. This means the glyph chart has limited space, which, in the ideal world, would be almost entirely spent on the glyphs, to provide the most in-depth comparison between glyphs. Secondly, comparing different values of confidence needs to be as easy as possible. To further improve upon the purely numerical values in candidate 1, two visual candidates were considered. The first candidate uses circles to convey the confidence information to the user. This candidate, when compared to candidate 3, is a lot more space efficient, but doesn't provide the same easy comparison. For the implementation of the tool, the choice was made that including a more easy comparison was more important than being space-efficient. This means that candidate 3 was used in the final implementation.

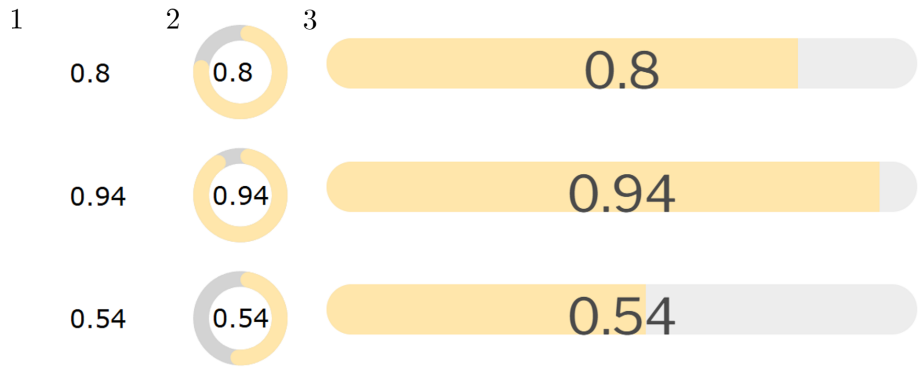


Figure 4.5: Candidates for confidence visualization in the glyph chart. Candidate 3 was deemed the most readable, thus being chosen for the final implementation.

Found Pattern Matching Algorithm To find the combinations of found patterns and ground truths Algorithm 1 was used. This algorithm compares the start and end of the ground truth to the start and end of the found pattern. Importantly, this algorithm does not compare the actual values of the time series, but only the temporal location of the start and end of the

ground truth or found pattern. Two cases exist for matching a found pattern to a ground truth. In the first case, either the start, end or both the start and end of the found pattern fall within the ground truth. This is covered by the first two conditions within the algorithm. The last condition covers the case that the found pattern starts before the ground truth and ends after the ground truth. This would result in the found pattern covering the ground truth, but is still a match for the ground truth. This results in a matching of ground truths with found patterns, where one ground truth can be matched with multiple found patterns. To store this matching, a *glyph* object was defined, which stores a single ground truth and a single found pattern. In the case a ground truth is matched with multiple found patterns, the ground truth will be added to the list of glyphs multiple times in different glyphs.

Algorithm 1: Glyph matching algorithm: getMatches()

Input : groundTruths (list of all predetermined ground truths)
Input : foundPatterns (list of similar patterns found by pattern search algorithm)
Output: glyphs (list of matching found patterns and ground truths)

```
1  glyphs = []
2  foreach groundTruth do
3      foreach foundPattern do
4          if groundTruth.start ≤ foundPattern.start ≤
            groundTruth.end then
5              glyphs.add(new glyph(groundTruth, foundPattern))
6          end
7          else if groundTruth.start ≤ foundPattern.end ≤
            groundTruth.end then
8              glyphs.add(new glyph(groundTruth, foundPattern))
9          end
10         else if foundPattern.start ≤ groundTruth.start and
            foundPattern.end ≥ groundTruth.end then
11             glyphs.add(new glyph(groundTruth, foundPattern))
12         end
13     end
14 end
```

Identifying False Positives and False Negatives Algorithm 1 gave us a list of glyphs, which individually match ground truths to found patterns. This algorithm only finds matches, which means that only true positives are

found. Luckily, the same matching logic can be used to check if a ground truth has any matching found patterns or if a found pattern has any ground truths. In the case the ground truth or found pattern does not have a match in the data, they are added as false negatives or false positives respectively. After being identified, these false negatives and false positives are added to the list of glyphs, with either the found pattern or ground truth being undefined.

Filtering Found Patterns When combining the choices made for visualizing the overlap within glyphs and the best colors, a sorted glyph chart is introduced into the approach. The final version of the glyph chart can be seen in Figure 4.1.A. This visualization shows users where found patterns are in comparison to their matching ground truths, combined with the confidence of the found pattern. The glyph chart also shows the currently selected confidence cutoff. This confidence cutoff, which removes all found patterns with a confidence lower than this cutoff, allows users to only look at glyphs with particularly high confidence. Being able to select a confidence cutoff is a good reflection of the real-world use of a time series pattern search algorithm, as only patterns that meet a minimum confidence requirement would be considered matches in most real-world applications. The confidence cutoff can be selected in a different visualization, which shows the distribution of the confidence within the data. More detail on this interaction can be found in the explanation of the confidence histogram in Section 4.2.2.

Glyph Sorting The main section of the glyph chart shows the visualized glyphs in order. Initially, the glyphs are sorted according to their occurrence in the time series, with glyphs starting earlier being higher in the glyph chart. Additional sorting methods can be selected in a drop-down menu above the glyph chart, as can be seen in Figure 4.1.A. The available sorting methods include *chronological ascending*, where glyphs that start earlier are shown higher, *chronological descending*, where glyphs starting later are shown first, *confidence ascending*, where glyphs that contain a found pattern with higher confidence are shown first and lastly *confidence descending*, where glyphs that contain a found pattern with lower confidence are shown first. While false positives can be sorted in the same way as regular glyphs, false negatives do not have a confidence value. The choice was made to sort the false negatives to the back of the list, displaying them as the last glyphs in the glyph chart.

4.2.2 Confidence Histogram

To select a confidence cutoff for the glyph chart, the choice was made to include a histogram of the distribution of the confidence of found patterns for the used algorithm. An example of this confidence histogram is shown in Figure 4.1.B. Within this histogram, users can move the vertical black line to select a confidence cutoff for the glyph chart. This allows users to filter data in the glyph chart, which is explained in Section 4.2.1, on only glyphs with sufficiently high confidence, enabling the evaluation of glyphs that, in real-world applications, meet the confidence requirement to be considered a real match. The currently used cutoff is shown at the top of the histogram, as well as at the top of the glyph chart. When hovering over the histogram, a muted potential new cutoff line follows the mouse. The numerical value associated with this line is shown next to the currently used cutoff at the top of the visualization, allowing for a very precise selection. To provide feedback on which parts of the distribution are part of the current selection, the opacity of bars that are not part of the current selection is lowered. Lowering the opacity of the bars that are not selected provides a similar result as highlighting the selected bars. In the case a glyph does not meet the requirements for confidence cutoff, the found pattern gets removed from the glyph. If the glyph contained a ground truth as well, this ground truth becomes a possible false negative. This possibility is checked automatically, and, if the ground truth does not match any other found patterns, the ground truth is added as a false negative to the glyph chart.

4.2.3 Time Series Graph

To accompany the glyph chart, which provides a more abstract visualization of ground truths and their matching found patterns, a line graph was implemented to give context to the glyph chart. This visualization will provide more information to the user, by showing the actual values of the time series as well as ground truths and found patterns. An overview of the final implementation of the line graph can be seen in Figure 4.1.D. The line graph includes the option for displaying multiple variables within the same time series, a mini-map to zoom into specific regions of the time series, and interactivity with the glyph chart and histogram. The different parts of this visualization will be explained further below.

Multivariate Line Graph The main area of the line graph shows the values of multiple selected variables over time. This is purely a real-world view of a time series, including different coloration for ground truths and colored

boxes for the found patterns. To further link the line graph to the glyph chart, the coloration of these ground truth lines and found pattern boxes is the same as in the glyph chart. Additionally, when the line graph is zoomed out entirely, the numerical identifiers, which are also included in the glyph chart, are included in the line graph. Because the line graph can become very crowded when more variables are included in the visualization, a variable or track selector was included in the implementation of the tool. This selector can be seen in Figure 4.1.C. To interact with this selector more easily, users can select multiple tracks at once by using the control or command key and the shift key to select their desired combination of tracks. When the user hits the save button, the line graph zooms out to the full-time series and shows the new selection of tracks.

Mini-map To further allow users to view only the parts of the time series that they deem interesting, a mini-map was included at the top of the line graph. This mini-map shows the current zoom level as a gray box together with all ground truths and found patterns as lines, where the dot shows the middle of the ground truth or found pattern. By dragging the lines at the edge of the gray box, users can resize the area that is shown in the line graph. Additionally, users can grab the box in its entirety and move it around the mini-map. This allows users to view different parts of the time series, without losing their comfortable zoom level.

View Coordination Lastly, the line graph is fully interactive with the other visualizations in the tool. When a user selects a glyph in the glyph chart that they think contains valuable information, they can click the glyph to get a more detailed view within the line graph. When this occurs, the line graph zooms into the temporal location of that glyph, maintaining a little padding at the front and back to better place the glyph into the time series. Clicking on any glyph while the line graph is zoomed in resets the zoom level of the line graph, again showing the entire time series. This is also reflected in the mini-map, where the new zoom levels are shown. An example of the line graph showing a detailed and zoomed view of a single glyph is shown in Figure 4.6.

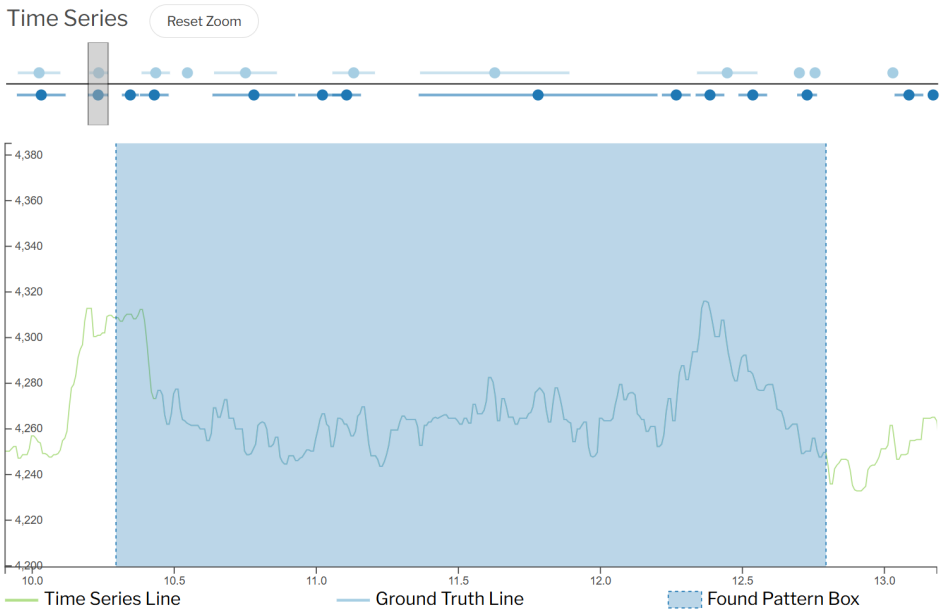


Figure 4.6: Time series line graph, zoomed into a single glyph

Filter Interactions When selecting a confidence cutoff in the histogram, the line graph also eliminates any found patterns that do not meet the minimum confidence requirement to be shown in the glyph chart. This further establishes the line graph as a visualization, serving the purpose of providing context for the information in the glyph chart. When making a selection in the scatter plot, which will be explained further in Section 4.2.4, the numerical identifiers of the selected glyphs will also be included in the line graph, again linking the line graph to the other visualizations.

4.2.4 Glyph scoring

Scatter Plot The final visualization of the algorithm exploration module can be found in Figure 4.7. This scatter plot uses two new values to plot each found pattern. *Ground Truth Coverage Percentage* is depicted on the x-axis. This shows the percentage of the ground truth which is covered by the found pattern. The y-axis depicts the amount of excess that is found in a found pattern. Excess is the amount of time contained in a found pattern that is not matched with a ground truth. How the excess and ground truth coverage percentage are calculated is explained in the next section, along with the limitations of the current algorithms. According to these values,

patterns are placed inside the scatter plot, which allows users to quickly evaluate the quality of a found pattern. Due to the nature of the way the depicted values are calculated, only true positives can be found in the scatter plot. The choice was made to allow the user to alternate between using the confidence histogram or the scatter plot, as these are both used for filtering data. Users are able to select which filtering method they want to use at which point, by changing the visualization shown using the tabs, as can be seen in Figure 4.1.B.

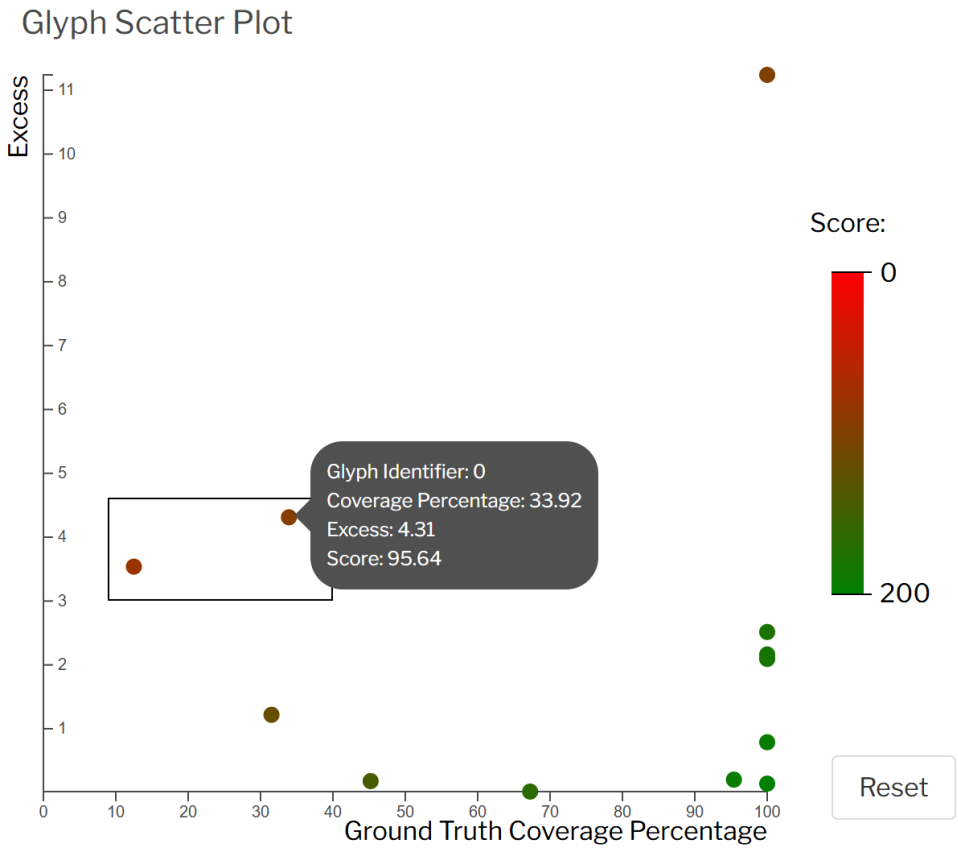


Figure 4.7: Scatterplot, visualizing scores for all glyphs

Score Visualization To reinforce the quality visualization of found patterns, the choice was made to calculate an actual score for each found pattern. This is visualized in the scatter plot by using the color of each found pattern dot. A reference color gradient can be seen on the right side of the

plot. Higher-quality found patterns are shown in green, while lower-quality found patterns are shown in red. Due to the nature of the used axes, the coloring will follow a gradient from the top-left to the bottom-right of the plot. The reference gradient follow roughly the same direction, as opposed to showing higher scores higher in the page. How this score is calculated will be explained in the next section, along with the explanations of the calculations of excess and ground truth coverage percentage.

Found Pattern Scoring To further identify which found patterns are most valuable to explore, found patterns are given a score. This score is used to numerically measure the quality of the pattern according to relevant metrics. Scored found patterns allow users to only look at glyphs scoring particularly high or low, enabling a more detailed evaluation of the workings of the used algorithm. The scoring algorithm, detailed in Algorithm 2, scores found patterns according to the same two metrics shown in the scatter plot, as explained in the previous section. How these metrics are calculated will be explained further following the explanation of the scoring algorithm. Found patterns are scored on a scale between 0 and 200. A higher score means a found pattern is of better quality according to these metrics. The score for a found pattern is calculated using the output of the excess calculation algorithm and the coverage percentage calculation algorithm. Both metrics are deemed equally important in this version of the approach. Because the coverage percentage is a number between 0 and 100, a scale is used to convert the absolute value of the excess for a glyph to a number between 0 and 100 as well. This scale maps an excess value to a number between 0 and 100, where 0 is the highest excess found within all glyphs and 100 is the lowest excess found within all glyphs. The result of the excess scoring scale is added to the coverage percentage to calculate the total score for a glyph, where a total score closer to 200 means the found pattern is deemed of better quality. To alter the importance of either metric, the result of the excess scoring scale can be altered. For example, giving the glyph with the lowest excess only 50 points would result in the coverage percentage being deemed twice as important for the total score a glyph will receive.

Algorithm 2: Glyph scoring algorithm

Input : glyph (one glyph)
Input : scoreScale (maps values between the highest and lowest excess to a value between 0 and 100)
Output: score (a score between 0 and 200)

```
1 score = 0
2 excess = getExcess(glyph.groundTruth, glyph.foundPattern)
3 excessScore = scoreScale(excess)
4 coveragePercentage =
   getCoveragePercentage(glyph.groundTruth,
   glyph.foundPattern)
5 score = excessScore + coveragePercentage
```

Excess and Ground Truth Coverage As can be seen in Algorithm 2, all glyphs found in the list of glyphs, generated by Algorithm 1, are scored in two steps. The first step calculates the excess of the algorithm and calculates a score between 0 and 100, using a scale of all scores in this set of data. The second step calculates the coverage percentage. After completing these two steps, these scores are added together to result in the final score for a found pattern. Next, the calculation algorithms for the excess and coverage percentages are explained.

Excess Calculation Firstly, the excess data is calculated in Algorithm 3. When given a matching ground truth and found pattern, the algorithm calculates the excess data on both sides. In the case the found pattern contains excess data at the beginning of the pattern, the start of the found pattern will be smaller compared to the start of the ground truth. The algorithm calculates the amount of excess by subtracting the start of the found pattern from the start of the ground truth. Similarly, if the found pattern contains excess data at the end, the end of the found pattern will be larger when compared to the end of the ground truth. The excess on this side is calculated by subtracting the end of the ground truth from the end of the found pattern. This means that when either of these values is positive, there is excess value on that side. When either of these values is negative, the found pattern does not cover the entire ground truth.

Algorithm 3: Excess calculator: getExcess()

Input : groundTruth (one ground truth)
Input : foundPattern (one found pattern)
Output: excess (2-tuple containing excess left and excess right)

```
1 excess = []
2 excessLeft = groundTruth.start - foundPattern.start
3 excessRight = foundPattern.end - groundTruth.end
4 excess.add(excessLeft)
5 excess.add(excessRight)
```

Coverage Percentage Calculation Secondly, the coverage percentage of the found pattern is calculated in Algorithm 4. This algorithm again takes a matching ground truth and a matching found pattern as input. Using the previous Algorithm 3, the excess on both sides is used to identify where the pattern has excess data. After this, the length of the ground truth is calculated, to later be compared to the found pattern. In the case the found pattern has excess data at the start, or Algorithm 3 returns 0 for the start, the found pattern starts before the ground truth or at the same time. Similarly, if the found pattern has excess data at the end, or Algorithm 3 returns 0 for the end, the found pattern ends after the ground truth or at the same time. The first condition covers the case that the found pattern has excess data on both sides. This would mean the found pattern at least entirely covers the ground truth, resulting in a coverage percentage of 100 percent. The next two conditions cover the cases in which either the excess at the start or the excess at the end is negative. This means the algorithm covers one side but, on the other side, it does not entirely cover the ground truth. The conditions consider on which side the found pattern does not entirely cover, to then calculate the percentage using the start and end of either the ground truth or found pattern. Using these values, the length of the found pattern within the ground truth is calculated. This length is divided by the total length of the ground truth and converted to a percentage value, to be used in the final scoring algorithm. In the final case, the found pattern has negative excess on both sides. This means both the start and end of the found pattern fall within the ground truth, resulting in the only part of the ground truth covered by the found pattern being exactly the found pattern itself. The length of the found pattern is divided by the total length of the ground truth and converted to a percentage.

Algorithm 4: Coverage percentage calculator: getCoveragePercentage()

Input : groundTruth (one ground truth)
Input : foundPattern (one found pattern)
Output: coveragePercentage (percentage of ground truth covered)

```
1  coveragePercentage = 0
2  excessLeft = getExcess(groundTruth, foundPattern)[0]
3  excessRight = getExcess(groundTruth, foundPattern)[1]
4  groundTruthLength = groundTruth.end - groundTruth.start
5  if excessLeft ≥ 0 and excessRight ≥ 0 then
6  |  coveragePercentage = 100
7  end
8  else if excessLeft < 0 and excessRight ≥ 0 then
9  |  coveragePercentage = (groundTruth.end -
   |   foundPattern.start) / groundTruthLength * 100
10 end
11 else if excessLeft ≥ 0 and excessRight < 0 then
12 |  coveragePercentage = (foundPattern.end -
   |   groundTruth.start) / groundTruthLength * 100
13 end
14 else if excessLeft < 0 and excessRight < 0 then
15 |  coveragePercentage = (foundPattern.end -
   |   foundPattern.start) / groundTruthLength * 100
16 end
```

4.3 Algorithm Comparison

The last functional module in the implementation is the *Algorithm Comparison* module. This module was designed to give users the ability to directly compare the results of different algorithms in the same section of the system. While the exploration tool technically allows for the comparison of multiple algorithms, using separate instances of the tool for separate algorithms, being able to directly compare different algorithms would provide a lot more information to users. The creation of this module relied heavily on decisions and algorithms made for the *Algorithm Exploration* module, which are used in altered or new visualizations. Due to time constraints, this module is more limited in its functionality compared to the *Algorithm Exploration* module. These limitations will be discussed further in Section 6.1. An overview of the

Algorithm Comparison module can be found in Figure 4.8, below. Again, this overview uses letter labels for further reference to the overview.

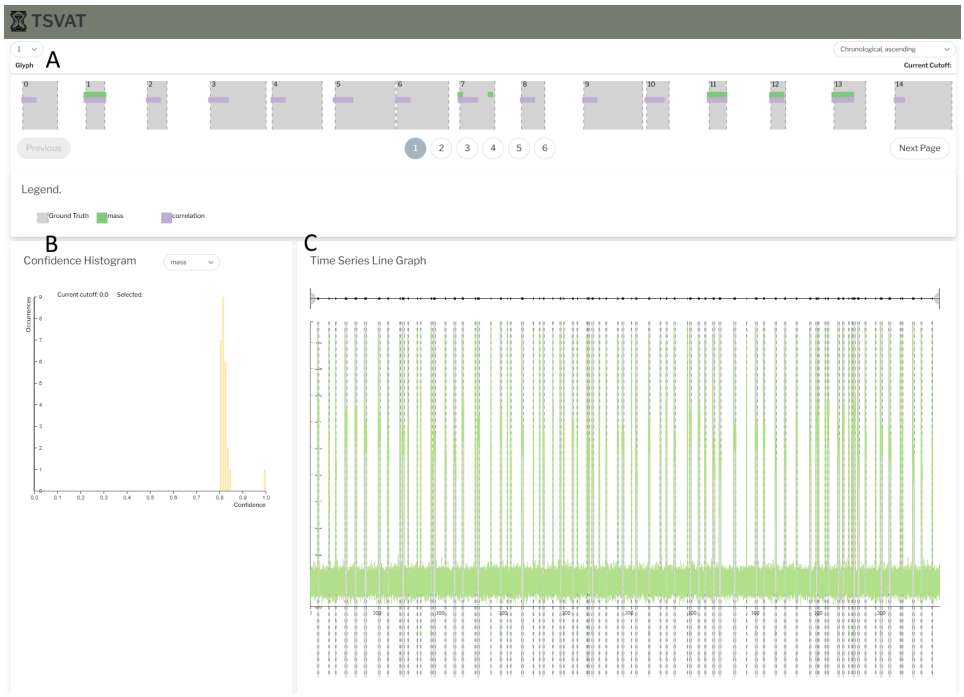


Figure 4.8: An overview of the *Algorithm Comparison* module. A larger version of this figure is available in Appendix B.

Module Overview As can be seen in Figure 4.8, the three main components explained in Section 4.1 are again present. Labeled with the letter A, the glyph chart is again the primary visualization for this module. The histogram, showing the confidence distribution for one of the algorithms, is labeled B. A line graph, showing the entire time series, together with found patterns and ground truths is labeled C. These visualizations will be explained in the next sections.

4.3.1 Glyph Chart

Similar to the algorithms exploration part of the tool, the comparison of algorithms is based on glyphs. Because a single ground truth can be matched with as many found patterns as there are algorithms to be compared, the simpler implementation detailed above does not suffice anymore. Using an

altered version of a glyph list, which will be explained more in Algorithm 5, and the lessons learned from creating the algorithm exploration part of the tool, a glyph chart was created. This glyph chart uses similar visualization techniques compared to the exploration glyph chart, but combines multiple found patterns with a single ground truth, to compare the output of different algorithms. An example of this glyph chart can be seen in Figure 4.8, labeled with the letter A.

Horizontal Glyph Chart To account for the variable number of algorithms that can be compared in the visualization, the choice was made to change to a horizontal version of the glyph chart. This choice also provides benefits for comparing found patterns inside glyphs. Comparing the relative position of different found patterns becomes easier by visualizing them this way. Due to the nature of the new glyphs, which show more found patterns for a single ground truth, the choice was made to not implement the line graphs within glyphs. Not having line graphs within all glyphs allows the focus to be more on comparing algorithms, reinforcing the abstraction from the time series data. An example of two glyphs within the glyph chart can be found in Figure 4.9.

Figure 4.9: Two glyphs within the comparison glyph chart

Vertical Ground Truth Visualization To compare multiple found patterns linked to a single ground truth, the choice was made to visualize the ground truth as a box, which spans the entire height of the visualization. This gives vertical space to show multiple algorithms' worth of found patterns. In Figure 4.9, this can be seen as a green and a purple rectangle covering the ground truth. The relative temporal locations of these rectangles provide the essential comparison to the user, allowing users to see which algorithms result in which accuracies, inaccuracies, excess and coverage.

Glyph Sorting The comparison glyph chart allows users to sort glyphs

similarly to the exploration glyph chart. The chronological sorting that was implemented for the exploration glyph chart is still included in this chart, as well as some new sorting options based on the confidence values of the found patterns within each glyph. All sorting options can be used in both an ascending and descending form. The first new option, *Average Confidence*, allows users to sort the chart based on the average confidence of all patterns within each glyph. Next, *Highest Confidence* and *Lowest Confidence* give users the option to sort based on the absolute highest and lowest confidence found within each glyph. Lastly, *Confidence Discord* provides the option for sorting glyphs based on the difference between the absolute highest and absolute lowest confidence in each glyph.

Legend Below the glyph chart, a legend is generated. This legend automatically creates visual references for all algorithms that are to be compared, as well as the color that is used for that algorithm. The colors used are again consistent over different visualizations in the same part of the tool.

Algorithm 5: Glyph matching algorithm for method comparison

```
Input  : groundTruths (list of all predetermined ground truths)
Input  : methods (list of all algorithms to be compared)
Output: truthMatching (list, mapping found patterns for all
        algorithms to matching ground truths)
truthMatching = new Map
foreach groundTruth do
    falseNeg = true
    foreach methods do
        currentMatches = getMatches(groundTruth, method)
        if currentMatches.length > 0 then
            falseNeg = false
        end
        truthMatching.add([groundTruth,currentMatches]
    end
end
```

Matching Multiple Found Patterns While the matching of ground truths and found patterns is the same as previously described, a new algorithm is implemented that does the matching for every included algorithm. This new algorithm can be found in Algorithm 5. As can be seen in Algorithm 5, the result of the algorithms is a Map. To best describe this object type, a Map

holds pairs of keys with corresponding values. The keys used in this Map are all ground truths found in the provided data. While not every pattern is aware of the algorithm they belong to, this does need to be stored in the matching, to allow for proper visualization. To achieve this essential link, the value in the *truthMatching* Map, which is the result of Algorithm 5, is also a Map. In this nested Map, the keys are all the algorithms that are to be compared, with the value being all found patterns that match the key of the truthMatching map. A more visual representation of an example of this structure is found in Figure 4.10.

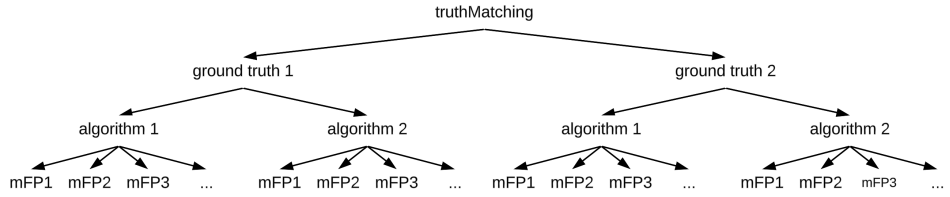


Figure 4.10: An example structure of a truthMatching Map. mFP stands for matching found pattern. In practice, truthMatching maps often contain many more ground truths and algorithms.

4.3.2 Line Graph

Mini-map zooming Similarly to the exploration part of the tool, the comparison also uses a line graph to provide more information on the results of time series pattern search algorithms. This line graph can be seen in Figure 4.8, with label C. The line graph struggles with longer time series, as can be seen in the figure. To make sure the line graph provides benefit to users, two interactions used for the exploration line graph were used. Firstly, the line graph again includes a mini-map at the top. This implementation of the mini-map is simplified when compared to the previously explained version. Because of the variable number of algorithms that can be compared, it is hard to determine how much space the mini-map would require to show an unknown amount of found patterns. The choice was made to only show the ground truths, visualized as bold parts of the line. This also allowed a simplified implementation of the current selection as well. In this case, the current selection is shown as two circles, where the left one shows the start and the right one shows the end of the selection. These circles can be dragged by the user to create the wanted zoom level.

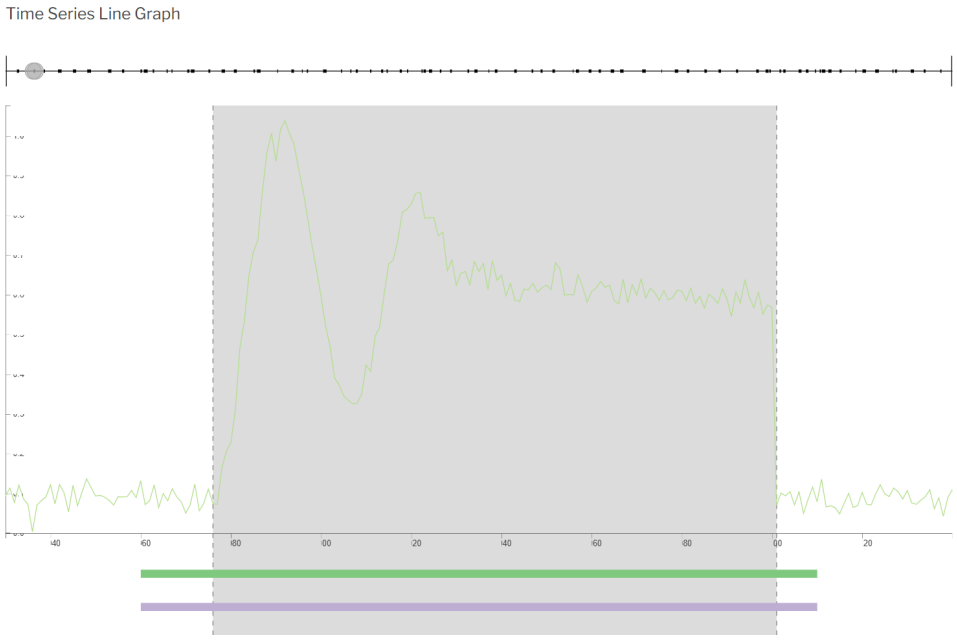


Figure 4.11: Comparison line graph, zoomed in on a single glyph

Zooming by Context Secondly, the user again can use the glyph chart to zoom. When a glyph is clicked in the glyph chart, similar behavior occurs as was the case in the *Algorithm Exploration* module, which is detailed in Section 4.2.3. The currently selected glyph is highlighted, while the line graph zooms into this glyph. To highlight the selected glyph, the background of the glyph is made darker. An example of a zoomed line graph can be seen in Figure 4.11. The mini-map again reflects the zoom, even when changed by the glyph chart. In this case, the two circles that are used to change in the mini-map are covering each other, as this time series contained a large amount of data.

Found Pattern Visualization As can be seen in Figure 4.11, the ground truth and found patterns are visualized in the same way as they are in the glyph chart, as opposed to the line graph in the *Algorithm Exploration* module. Rectangles below the x-axis of the line graph show which parts of the time series are found by which algorithm, using the same colors as they do in the glyph chart. The choice was made to not directly show the found patterns as part of the line graph due to the comparison between algorithms being the primary focus of this module.

4.3.3 Confidence Histogram

The final visualization is the confidence distribution histogram, which can be seen in Figure 4.12 This behaves the same way as the version in the *Algorithm Exploration* module. Again, this histogram shows users the current confidence cutoff and allows users to filter found patterns based on a selected confidence cutoff. Selection of a cutoff higher than the confidence for a found pattern removes that found pattern from its corresponding glyph. Other found patterns for the same glyph remain visualized as long as their confidence is higher than the selected confidence cutoff. This version however also includes a drop-down menu at the top, which allows users to select which algorithm they want to be visualized in the histogram.

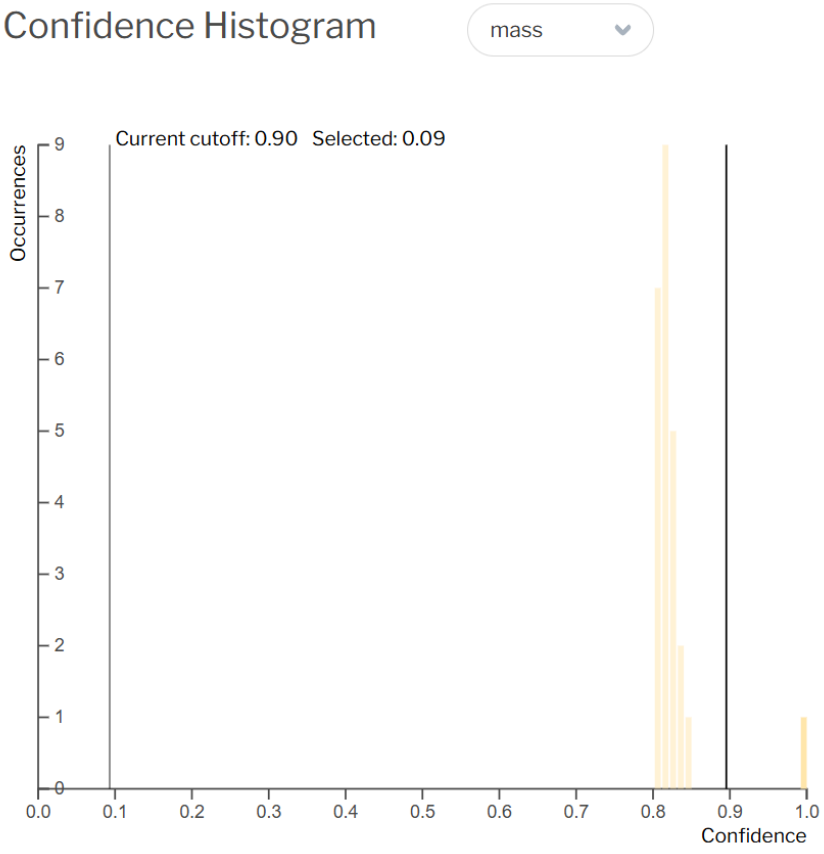


Figure 4.12: Comparison line graph, zoomed in on a single glyph

4.4 Algorithm Version Comparison

While the original planning contained plans for a separate part of the tool for comparing relevance feedback rounds, due to time constraints the choice was made to include this in the algorithm comparison. Although the aim when comparing relevance feedback rounds is very different from comparing different algorithms, the visualizations used can be very similar. By viewing different rounds of relevance feedback as different algorithms, they can be used as input for the algorithm comparison tool. This results in a comparison, which could look the same as comparing different algorithms, but with a different purpose.

Chapter 5

Evaluation

This chapter aims to prove the added benefit of using the proposed visualization approach when evaluating time series pattern search algorithms. To measure this added benefit, two real-world scenarios are going to be studied in separate case studies. The two modules of the system are going to be evaluated using separate case studies, using the tasks detailed in Chapter 3. Each module has a subset of tasks that are supposed to be completed using that module. Those tasks will be used to create the structure of the case studies, following the tasks and subtasks in sequential order where possible. First, the data used in these case studies is going to be explained, to give some domain context to the evaluation in the case studies. After the explanation of the data, two case studies are described, each reflecting a real-world scenario where users utilize the implementation of the visualization approach to do research on data. The next section of this chapter provides an overview of the existing evaluation methods, as described in Chapter 2, trying to find the added benefit of the visualization approach within the state-of-the-art. Together with Section 5.3, the case studies will highlight a practical application of the visualization system, demonstrating its unique potential to add to the state-of-the-art in evaluating time series pattern search algorithms. The chapter will end by explaining an expert study performed on the implementation of the visualization approach, along with a summary of the results of this expert study.

5.1 Data

The first case study was conducted using the *Algorithm Exploration* module of the visual system. The data used in this case study was conducted using

the EEG Eye State dataset [Roesler, 2013]. To create this dataset, a participant was asked to periodically open and close their eyes. The measurement was gathered using an electroencephalography (EEG) neuroheadset, which measured electrical activity in the brain in 17 different locations on the head. While the experiment was going on, the participant was filmed opening and closing their eyes. This video footage allowed for the comparison of the measured data with the video footage, annotating the exact times the subject's eyes were open and closed. Annotating the data generated clear ground truth data to allow pattern search on this dataset. The dataset consists of 14.980 instances of data, over a duration of 117 seconds. No missing values are found in the 14 features of the dataset.

For the second case study, the *Algorithm Comparison* module was used. The Deep Valve dataset used in this recorded the electric current running through a solenoid valve while being tested in a cooling systems test bench. Ground truths in this dataset contain a complete operating cycle for the solenoid valve. This operating cycle consists of up to five phases, starting with a peak in current when the valve is enabled. In the second phase, a clear "J" shape can be found in the data, followed by the movable part of the valve moving entirely from one end to the other. This dataset only contains a single variable, the electrical current measured. The dataset contains 100.000 instances of data, over 100 seconds of measurement, without any missing values. The Deep Valve dataset contains proprietary data and thus can not be published.

5.2 Case Studies

5.2.1 Case Study 1: Algorithm Exploration Module

The *Algorithm Exploration* module, for which the design is explained further in Section 4.2, aims to provide the functionality to achieve the tasks in **T1**, **T2** and **T3**. Similar to assumptions made when analyzing the tasks in Chapter 3, it is assumed the user possesses files in the proper file format for the tool, starting the case study right after uploading the proper files. This places the start of the case study right at the start of the actual evaluation of the results of the time series algorithm. The algorithm that will be evaluated in this case study is based on Dynamic Time Warping [Berndt and Clifford, 1994] and will try to find periods where the participant of the measurement had their eye closed, in the future called eye-closed periods.

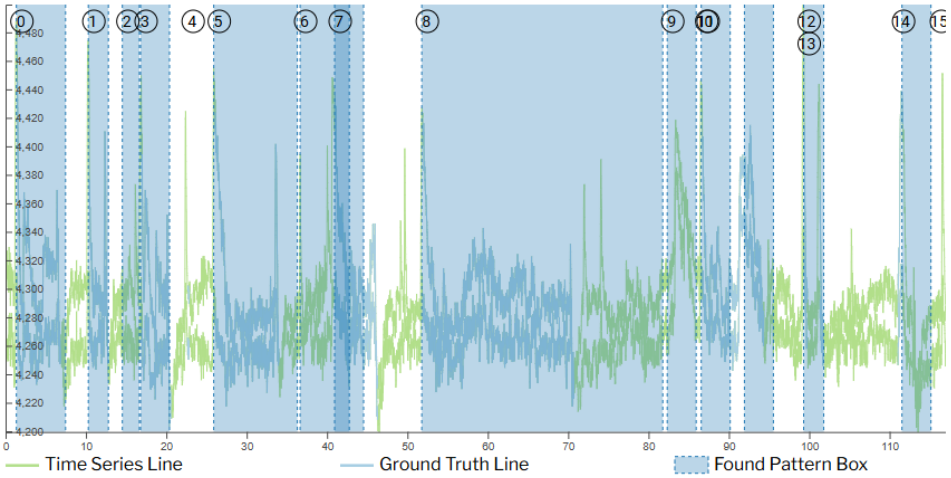


Figure 5.1: Initial line graph

After loading the proper data into the system, the analyst immediately noticed three main visualizations, as can be seen in the overview image in Figure 4.1. While looking at the page, the line graph at the bottom right drew their eye, shown in Figure 5.1. In this visualization, the analyst recognized familiar-looking patterns, recognizing them as some of the raw EEG data. The line graph showed the expected amount of data instances. Visualized over the line graph, the analyst found blue boxes and blue sections of the line. They immediately noticed that these boxes and line sections did not line up perfectly. They assumed these show the locations of expected eye-closed periods as found by the pattern search algorithm and predetermined actual eye-closed periods. This assumption was validated by the legend found under the line graph, found in Figure 5.1, showing the analyst which visual element was used for the eye-closed periods generated by the algorithm and the predetermined eye-closed periods.

At first analysis, the analyst found the algorithm to be on the aggressive side, seemingly finding the correct periods in the data, but often overshooting the end of the predetermined eye-closed periods. In Figure 5.1, the found patterns labeled 5, 8 and 14 show this overshooting behavior. Even some expected periods generated by the algorithm did not seem to have a corresponding predetermined period. With this first impression in mind, the analyst looked to the glyph chart for a more detailed analysis, because they were instructed the glyph chart is best suited for analyzing how well a generated found pattern in data matches a predetermined ground truth. Part of the glyph chart can be seen in Figure 5.2. Due to the similar colors in

this visualization, the rectangles in this graph felt familiar enough to the analyst that they assumed the information they were meant to convey, again backed up by the legend below the graph. Navigating through the pages of the glyph chart, the analyst started thinking their initial analysis was too hasty because the periods generated by the algorithm found only part of the expected period in many cases. The glyphs labeled 6, 7, 10, 11 and 14 clearly showed the analyst cases where the algorithm only finds part of the predetermined eye-closed periods, most of which as can be seen in Figure 5.2. The analyst even noticed the glyph labeled number 4, which seems not to have an expected period generated by the algorithm, as opposed to glyphs 2, 9 and 15, which do not cover periods that were predetermined to be eye-closed periods. Glyphs 2, 4 and 9 can also be seen in Figure 5.2. Regarding the missing expected eye-closed period for glyph 4, the analyst concluded that the predetermined eye-closed period was too short to be detected by the algorithm. This resulted in a False Negative in the glyph chart, which did not have a confidence value to be visualized. Glyph 2 contained an expected eye-closed period which, on further inspection, looked a bit like the predetermined eye-closed periods in other glyphs, starting with a relatively high value before lowering down to a plateau. This resulted in the algorithm branding this a similar pattern, with relatively low confidence.

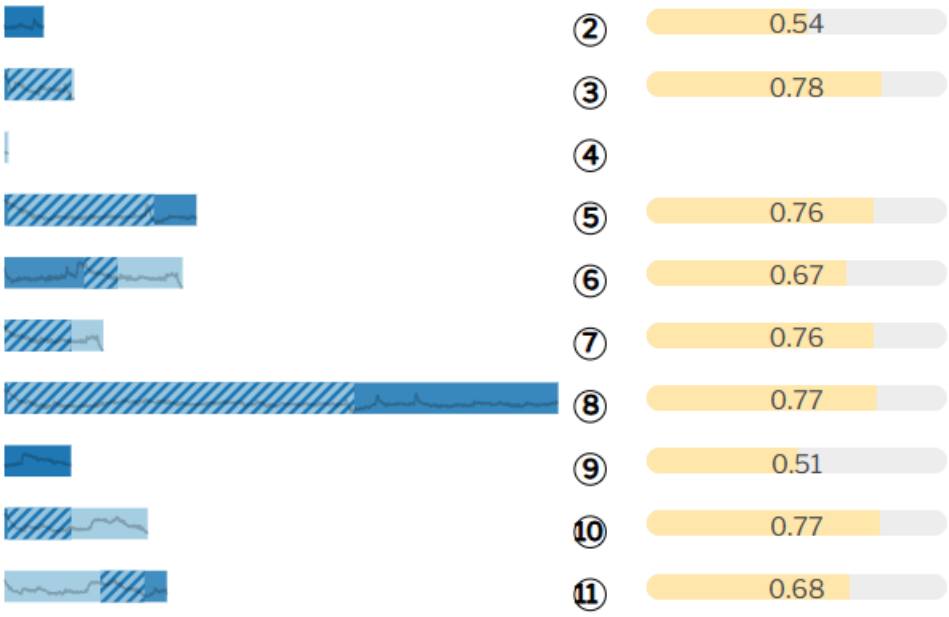


Figure 5.2: Part of the glyph chart

Figure 5.3: Glyphs 1 and 2, after sorting by Confidence descending

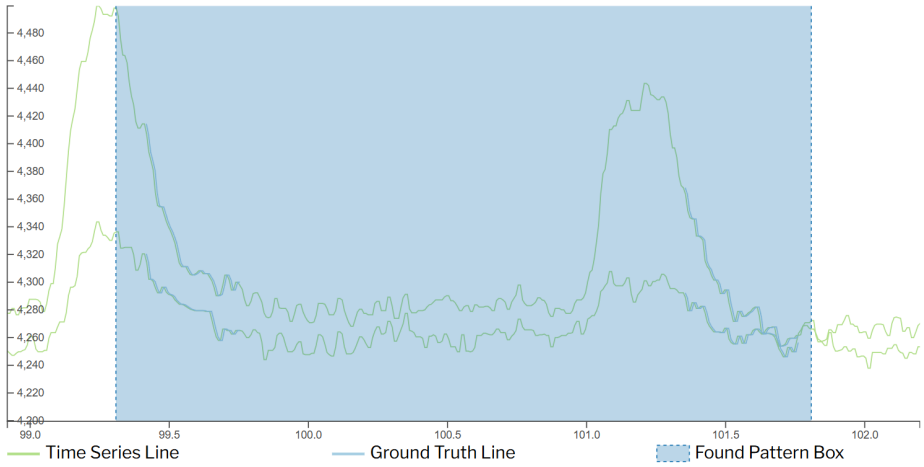


Figure 5.4: Zoomed line graph for glyph 1

After making these initial observations, the analyst desired to look deeper into patterns within the glyphs. By changing the sorting method of the glyph chart to *Confidence, descending*, the analyst was able to view the glyphs with high confidence at the top of the chart, providing the ability to more easily compare the results that the algorithm is most sure about. After noticing the identifiers of the glyphs change to a new order in the line graph, the analyst was able to identify which glyphs they wanted to have a look at next. With this new sorting, the analyst noticed very similar data for glyphs 1 and 2 in the glyph chart, as can be seen in Figure 5.3. By clicking on glyph 1, the line graph zoomed in to just before the start of the algorithm labeled eye-closed period in glyph 1, as can be seen in Figure 5.4. In this zoomed line graph, the analyst noticed that this algorithm labeled eye-closed period covered two predetermined eye-closed periods. As both these predetermined eye-closed periods were short, lacking the plateau that longer eye-closed periods and the query contained, the analyst concluded the algorithm probably had trouble finding shorter eye-closed periods or eye-closed periods which lacked the plateau ending.

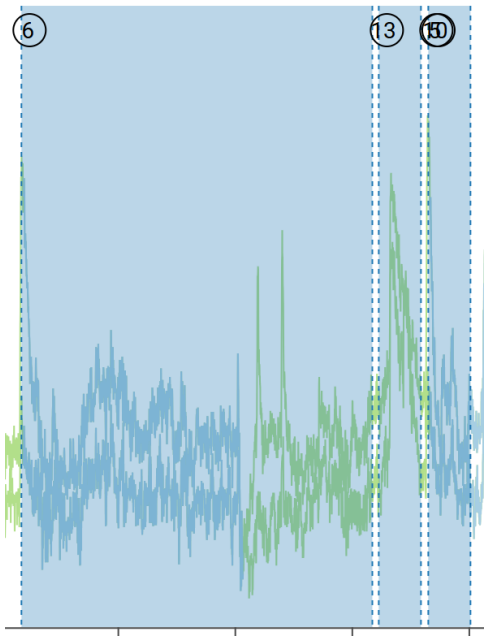


Figure 5.5: Part of the Line Graph, sorted by confidence descending

Figure 5.6: Zoomed line graph for glyphs 10 and 11

When zooming the line graph back out to the entire time series, the analyst noticed that the predicted eye-closed periods labeled 5 and 10 were overlapping, as seen in Figure 5.5. Wondering why this was the case, the analyst referred back to the glyph chart. First, they made sure to sort the glyphs back to chronological order, which grouped the two glyphs in question together. Cross-referencing the new identifiers within the line graph with the identifiers in the glyph chart, the analyst found that glyphs 10 and 11 covered the same predetermined eye-closed period. This was confirmed by the small line graphs within the glyphs, which can be seen in glyphs 10 and 11 in Figure 5.2, as well as zooming into either eye-closed period in the line graph, which can be seen in Figure 5.6. The analyst noticed similar cases before, where the algorithm recognized a single predetermined eye-closed period as two separate ones. This was likely because of a similar subpattern at the start and end of the predetermined eye-closed period, making this worthy of further research.

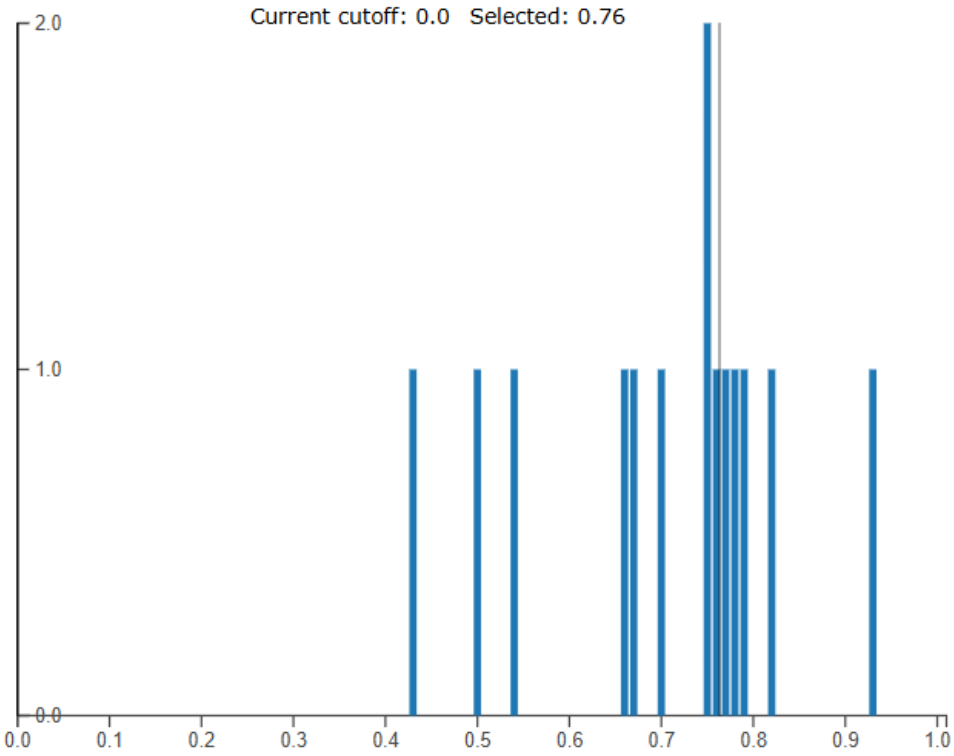


Figure 5.7: Confidence histogram, including median estimation

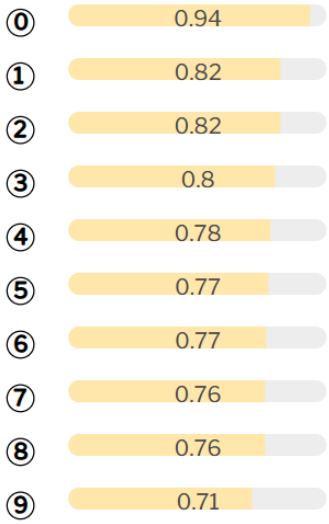


Figure 5.8: Descending confidence values for the first 10 glyphs in glyph chart

For their research, the analyst was only looking for predicted eye-closed periods that the algorithm was very sure about. When analyzing the confidence distribution histogram, as seen in Figure 5.7, along with the confidence values in the confidence sorted glyph chart, displayed in Figure 5.8, the analyst noticed seven of the eye-closed periods being identified with a confidence value above 0.76, while the other eight are below 0.76. The analyst concluded a confidence value above 0.76 to be higher compared to the median confidence value. To further their analysis, the analyst desired to filter the predicted eye-closed periods that did not meet this confidence requirement of 0.76. Prior instruction made the analyst aware that a confidence cutoff could be selected within the histogram, showing the distribution of confidences within the dataset of predicted eye-closed periods.

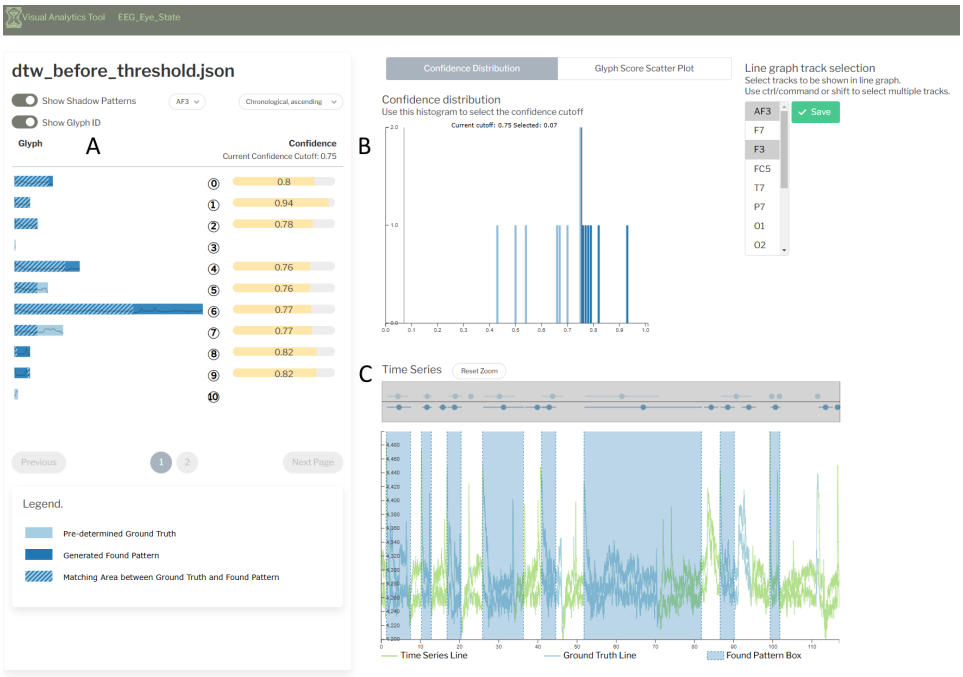


Figure 5.9: Zoomed line graph for glyphs 10 and 11

Configuring this confidence cutoff changed the visualized glyphs in the glyph chart, the confidence distribution histogram and the predicted eye-closed pattern visualized in the line graph. The result of this filtering can be seen in Figure 5.9. The amount of glyphs displayed in the glyph chart was reduced to 10, the analyst assumed false positives were removed from the visualization due to low confidence, as can be seen in Figure 5.9.A. The predicted eye-closed periods displayed in the line graph were also reduced, as can be seen in Figure 5.9.C. In Figure 5.9.B, the confidence histogram, the current cutoff was highlighted by a line, emphasized by the bars not included in this cutoff losing opacity.

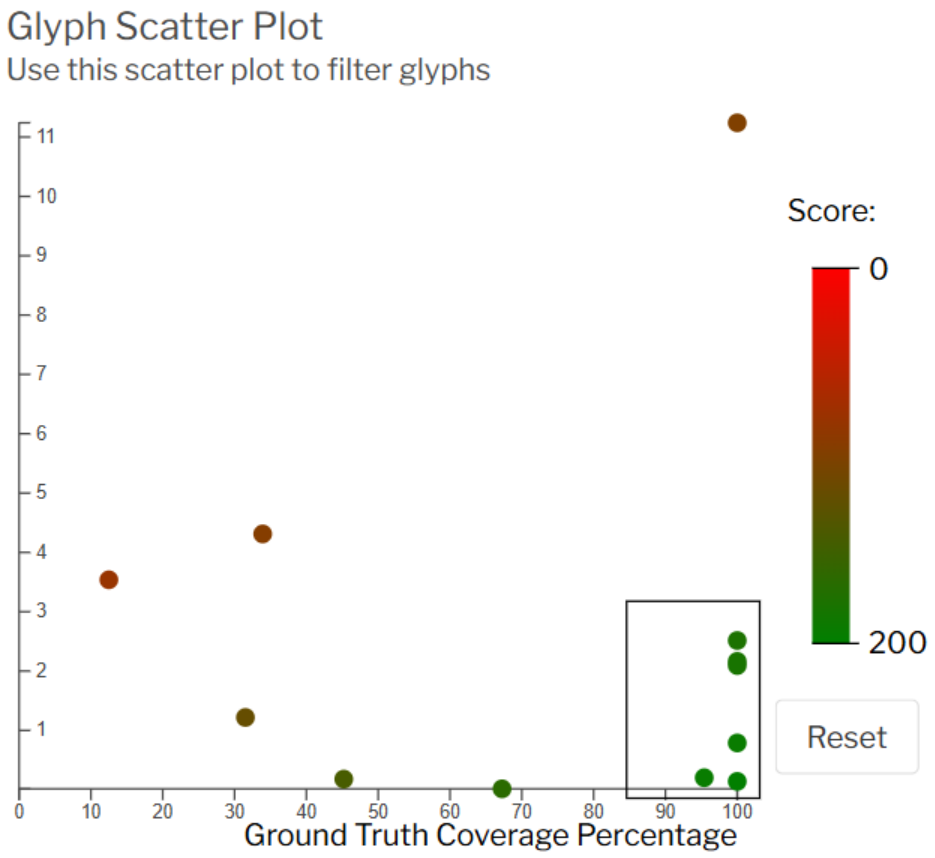


Figure 5.10: Scatter Plot for the results DTW-based based algorithm

Besides high confidence, the analyst wanted to know which predicted eye-closed periods were solid representations of the actual eye-closed period. To identify the highest-quality periods, the analyst was instructed to use the scatter plot. After changing the filtering visualization from the histogram to the scatter plot, the analyst viewed the information visualized in the scatter plot, which can be seen in Figure 5.10. According to the metrics used, the quality of predicted eye-closed periods was not very consistent. While six patterns scored pretty high in both metrics, the other predicted periods had flaws in one or both metrics. To identify why this is the case, he used a dragging motion to create a selection box to select only the patterns with a high score. The result of this dragging motion can be seen in Figure 5.10. This selection made the glyph chart only show the six selected glyphs. Immediately, the thoughts of the analyst were confirmed: the glyphs showed

comparatively small amounts of excess data compared to others, while still finding the entire predetermined eye-closed period. The analyst concluded that these algorithm-identified eye-closed periods were of good score due to clear, distinguishable features in these predetermined eye-closed periods. This was backed up by the relatively high confidence values of these identified eye-closed periods.

5.2.2 Case Study 2: Algorithm comparison

Tasks **T1**, **T2** and **T4** are realised in the *Algorithm Comparison* module, for which the specific design is detailed in Section 4.3. In this case study, two algorithms will be compared, one based on MASS [Mueen et al., 2022] and the other based on Correlation. These algorithms were tasked with finding points in the time series where the solenoid valve is activated, starts moving and ends moving. The entire process of moving the valve is called a valve operating cycle.

The analyst was greeted by three visualizations, as can be seen in the overview in Figure 4.8. Immediately, they noticed that the time series line graph was barely readable. Due to the large size of the used dataset, the line graph required zooming before analysis was possible. To achieve this, the analyst used the mini-map above the line graph to make the line graph only visualize part of the time series. This allowed the analyst to verify the uploaded data, noticing the similarity in the predicted operating cycles in the line graph, for both used pattern search algorithms. These initial observations made the analyst desire a more in-depth analysis of the results of both algorithms.

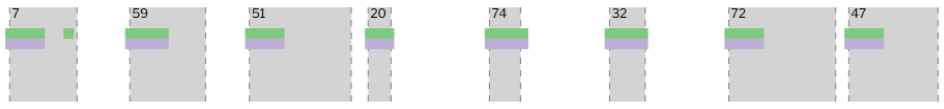


Figure 5.11: The first 8 glyphs in the glyph chart when sorted by average confidence

To achieve this more in-depth analysis, the analyst started by sorting the glyph chart on the average confidence of all patterns within a glyph, showing lower average confidences first. This was achieved by using the *Average Confidence, ascending* sorting option in the glyph chart. The 8 highest glyphs that they found in the sorted glyph chart can be seen in Figure 5.11. Glyph 7, shown to have the lowest average confidence, had a peculiar found

operating cycle for the MASS algorithm. To further investigate this glyph, the analyst clicked on it, which made the line graph zoom into the location of that glyph, which can be seen in Figure 5.12. In the line graph, the analyst found that both algorithms only seem to find the activation phase, which is the volatile period measured in the electrical current, when the solenoid valve loosens but is not yet physically moving. In addition to the activation phase, the algorithm based on MASS also found part of the later phases but identified it as a separate operating cycle. Using the pagination in the glyph chart, the analyst noticed a pattern, where most predicted activation cycles contain only the activation phase. The analyst hypothesized this phenomenon occurred due to larger fluctuations in electrical current in the penultimate phase of the operating cycle. This penultimate phase is the linear phase, where the solenoid valve physically moves from one side to the other.

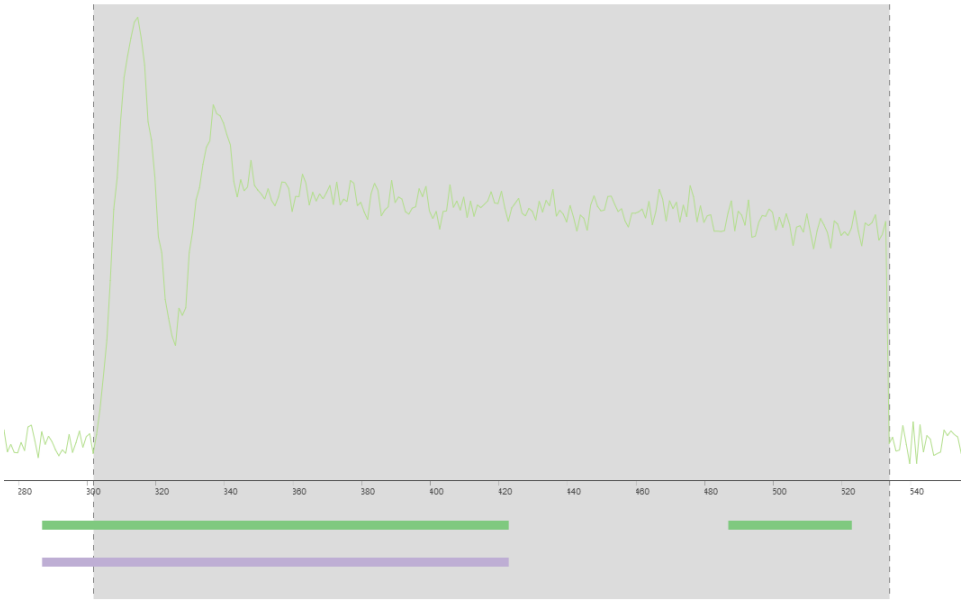


Figure 5.12: Line graph, zoomed into a single glyph

Navigating through the pages of the glyph chart also made the analyst notice a pattern in the average confidences for the glyphs. All glyphs containing a predicted operating cycle by the algorithm based on mass had lower average confidence compared to glyphs only containing predicted operating cycles by the algorithm based on correlation. To further analyze this,

the analyst looked at the confidence histogram. This confidence histogram visualized the distribution of confidences within each of the two different algorithms, based on a selection made in a drop-down at the top of the visualization. The confidence distribution for both algorithms can be found in Figure 5.13. When switching the histogram from displaying mass to displaying correlation, the analyst immediately confirmed his suspicions, noticing that the confidence for predicted operating cycles found by the correlation algorithm is distributed higher compared to those found by the mass algorithm. This was confirmed by interacting with the histogram. By selecting a confidence cutoff in the histogram between 0.85 and 0.9, almost all predicted operating cycles found by the mass algorithm were filtered from the glyph chart, leaving only the outlier with a higher confidence value and the predicted operating cycles found by the correlation algorithm.

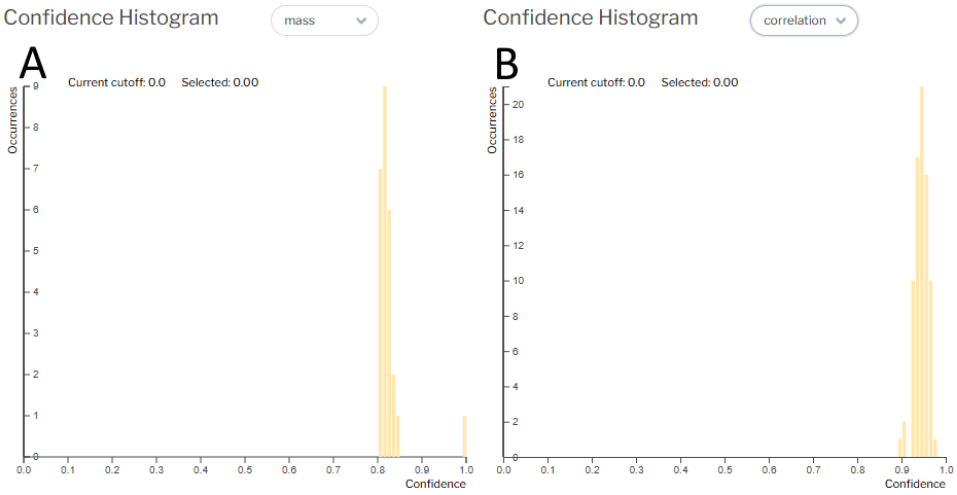


Figure 5.13: Confidence distributions for mass and correlation

5.3 Contribution to State of the Art

Due to visual evaluation tools for time series pattern search algorithm results being a largely unexplored field, comparing the proposed visual system to the state-of-the-art provides no additional insight into the benefit of this tool. While visual systems like TimeSearcher2 and PSEUDo, explained in Section 2.1.2, do provide ways of visualizing the results of time series pattern search algorithms, no tools were found that visualize predetermined ground truths as well as the results of time series pattern search algorithms. The

main novel visualization of the proposed visual system, the glyph chart explained in Section 4.2.1 and Section 4.2.1, contribute to the state of the art by developing a novel way of evaluating the results of time series pattern search algorithms, giving users the ability to directly compare the found patterns to predetermined ground truths.

5.4 Expert Study

The visualization approach proposed in this research was developed for a very niche audience. End users are mostly researchers developing time series pattern search algorithms. Due to the limited availability of end users, a single expert evaluation was conducted, without the expectation of doing statistical analysis as a result of this evaluation. The participant was a software developer, part of a team responsible for developing time series pattern search algorithms.

5.4.1 Structure

During the expert evaluation, the participant was first introduced to this research, the implemented visual system and the planned activities for the expert study session. The participant was encouraged to ask questions during the session, as having to ask questions provides information on the intuitiveness of the visualizations. After the introduction, the participant was given time to familiarise themselves with the visual system. When sufficiently familiar to complete analysis tasks using the visual system, the participant was asked to complete a list of tasks using the visual system. During the completion of these tasks, the participant was encouraged to think aloud. The final list of tasks completed in the expert study can be found in Section 5.4.1. Finally, to gather summarizing data on the user experience of the visual system, the expert study concluded with a questionnaire. The questions used in the questionnaire can be found in Section 5.4.1

Task List

The tasks in the expert study were grouped by which module they were to be completed with. Tasks marked by an asterisk were skipped due to accidental completion during prior tasks or during the system familiarization phase.

Algorithm Exploration Tasks

- Find the retrieved pattern with the lowest confidence
- Filter retrieved patterns below 0.7 confidence
- Zoom the line graph into a retrieved pattern to receive more detail on that pattern*
- Zoom into a specific subsequence of the time series in the line graph using the mini-map
- Sort retrieved patterns according to confidence, showing higher confidence patterns first*
- Select only retrieved patterns that received a good score in the scatter plot

Algorithm Comparison Tasks

- Zoom into a group of retrieved patterns in the line graph*
- Zoom into a specific subsequence of the time series in the line graph using the mini-map
- Compare confidence distributions for different algorithms in the histogram
- Filter retrieved patterns below 0.7 confidence

Questionnaire

To summarize the experience of using the visual system, the participant was asked to answer 13 questions about both modules of the visual system. Similar to the tasks, the questions are grouped by the modules of the visual system. Because no statistical analysis was intended to be conducted on the answers to the questionnaire, most questions were open-ended questions, encouraging more elaborate answers.

Algorithm Exploration Module

1. Did the algorithm exploration module meet your expectations?
2. What is, in your opinion, the most important element in the algorithm exploration module?

3. On a scale of 1 to 5, where 1 is really complicated and 5 is very easy, rate the difficulty of finding the features requested in the tasks within the algorithm exploration module.

4. Was it easy to make sense of the interactions between different visualizations?

5. How easy was it to find the information you required in the tool?

6. Do you think the algorithm exploration module provides additional opportunities for the evaluation of time series pattern search algorithms?

7. Would you use the algorithm exploration module again in the future?

Algorithm Comparison Module

1. Did the algorithm comparison module meet your expectations?

2. What is, in your opinion, the most important element in the algorithm comparison module?

3. On a scale of 1 to 5, where 1 is really complicated and 5 is very easy, rate the difficulty of finding the features requested in the tasks within the algorithm exploration module.

4. Was it easy to make sense of the interactions between different visualizations?

5. How easy was it to find the information you required in the tool?

6. Would you use the algorithm comparison module again in the future?

Results

While completing the list of tasks, the participant required more pointers than previously expected. When required, small pointers were sufficient to get the participant back on track to completing tasks. Regarding requiring these pointers, the participant remarked that it was mostly due to unfamiliarity with data analysis tools. Completing the tasks all went with relative ease. The participant even managed to complete later tasks while in the process of completing earlier ones. Notably, the participant immediately found patterns that interested them during the tasks, selecting them for zooming without any prior explanation. This resulted in them noticing that only part of the ground truth was found in this particular pattern, which was

a correct observation. Additionally, when tasked with finding the highest confidence pattern in the glyph chart, the participant immediately resorted to using sorting options to view the highest pattern first, in both modules. While not strictly part of the intended tasks, the participant even wanted to look at the sorting options specific to the *Algorithm Comparison* module, although these new ones did require an explanation before the participant understood their added benefit. Backed up by the answers to the questionnaire, which are included in Appendix C, the confidence cutoff functionality, found in the confidence distribution histogram in both modules, was not immediately clear to the participant, requiring specific explanation before the corresponding tasks could be completed.

The summarizing questionnaire resulted in very positive remarks on the system. As can be seen in the answers to the questionnaire in Appendix C, the participant stated both modules met the expectations set by the introduction to the system. Answering question number 2, for both modules, the participant remarked that the most important element in the modules was different between them. For the *Algorithm Exploration* module, the participant deemed the glyph chart the most important, while for the *Algorithm Comparison* module the sorting and filtering options were more important. When asked about the added opportunities of a visual evaluation tool compared to existing evaluation methods, the participant mentioned exporting data and saving visualization settings as important features for the future. Even in the current state, when asked if the participant would want to use the visual system in the future they wholeheartedly answered yes.

Chapter 6

Discussion and Conclusion

6.1 Limitations and Future Work

This chapter will provide a list of limitations to the current version of the visualization approach and its implementation. While the current versions were found to provide beneficial contributions to the state of the art of evaluating time series pattern search algorithms by visualizing their results, it is important to acknowledge limitations in this research.

During the creation of the proposed visualization approach and the implementation of that approach, only a single dataset was used per visualization module. Given more time, more datasets should be used to verify the added benefit of the approach and implementation. More datasets can not only be used to verify the added benefit of the visualization tool, but it would also ensure the implementation is not functional on just a single dataset.

In the future, the added benefit of the proposed approach could be verified by conducting a more large-scale user study, including more end-users compared to the expert study in this research. While this may prove difficult and time-consuming to set up, including the user experience of more end-users would be the best way to verify the usability and the added benefit of the approach and implementation.

Besides evaluating the current implementation, further development on the *Algorithm Comparison* module and the *Algorithm Version Comparison* module could increase the real-world applicability of the approach and implementation. While the *Algorithm Exploration* module provided novel visualizations and evaluation techniques for the results of time series pattern search algorithms, this research was not able to fully explore opportunities in the other modules due to time constraints.

While the current implementation of the visualization approach did use a CSS framework, combined with custom CSS rules, to create a visually pleasing layout for the different modules, proper web design may increase the usability and accessibility of the system. The current implementation does not properly scale visualization according to the size of the screen, inhibiting working with the system on, for example, mobile devices. Fixing this limitation would improve the system drastically, while also providing opportunities for better handling large time series.

6.2 Conclusion

This research has explored opportunities in the visualization and visual evaluation of the results of time series pattern search algorithms. These opportunities were backed up by literature research, which, together with a predefined list of goals for this research, became the base for task analysis on the visualization of time series pattern search results. By developing a visual analysis approach and implementing that approach, we aimed to provide novel ways of evaluating time series pattern search algorithms, beyond existing numerical metrics, contributing to the extraction of meaningful insights from time series pattern search results.

Using case studies, the different modules of the proposed visual system were evaluated, proving the added benefit of using a visual analytics approach to evaluate time series pattern search algorithms. These added benefits were emphasized in an expert study, where an expert completed several tasks using the visual analytics system and answered several questions about their experience with the visual system. Both the case studies and the expert study provided promising results regarding the implementation of the visual analytics approach.

In summary, this research contributed to the field of time series analysis by providing a novel method for evaluating time series pattern search algorithms along with an implementation of this method. Being able to better evaluate time series pattern search algorithms will lead to more scalable, adaptable, and overall better-functioning solutions for different data domains. Within applicable domains, being able to better identify similar patterns in time series data will have huge benefits. These benefits take different shapes, specific to each domain. In the future, further research into verifying the benefits of the proposed approach and visual evaluation of time series pattern search algorithms as a whole may reveal additional opportunities for enhancing the evaluation of time series pattern search algorithms.

Bibliography

Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

Paolo Buono, Aleks Aris, Catherine Plaisant, Amir Khella, and Ben Shneiderman. Interactive pattern search in time series. page 175, San Jose, CA, March 2005. doi: 10.1117/12.587537. URL `http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.587537`.

Michael Correll and Michael Gleicher. The semantics of sketch: Flexibility in visual query systems for time series data. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 131–140, Baltimore, MD, USA, October 2016. IEEE. ISBN 978-1-5090-5661-3. doi: 10.1109/VAST.2016.7883519. URL `http://ieeexplore.ieee.org/document/7883519/`.

Cynthia Brewer, Mark Harrower, Ben Sheesley, Andy Woodruff, and David Heyman. COLORBREWER 2.0. URL `colorbrewer2.org`.

Philipp Eichmann and Emanuel Zgraggen. Evaluating subjective accuracy in time series pattern-matching using human-annotated rankings. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, page 28–37, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450333061. doi: 10.1145/2678025.2701379. URL `https://doi.org/10.1145/2678025.2701379`.

Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *ACM SIGMOD Record*, 23(2): 419–429, June 1994. ISSN 0163-5808. doi: 10.1145/191843.191925. URL `https://dl.acm.org/doi/10.1145/191843.191925`.

Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, February 2011. ISSN 0952-1976. doi: 10.1016/j.engappai.2010.09.007. URL https://www.sciencedirect.com/science/article/pii/S0952197610001727.

Tak-chung Fu, Fu-lai Chung, Robert Luk, and Chak-man Ng. Stock time series pattern matching: Template-based vs. rule-based approaches. *Engineering Applications of Artificial Intelligence*, 20(3):347–364, April 2007. ISSN 09521976. doi: 10.1016/j.engappai.2006.07.003. URL https://linkinghub.elsevier.com/retrieve/pii/S0952197606001278.

Xueyuan Gong, Simon Fong, Jonathan H. Chan, and Sabah Mohammed. NSPRING: the SPRING extension for subsequence matching of time series supporting normalization. *The Journal of Supercomputing*, 72 (10):3801–3825, October 2016. ISSN 0920-8542, 1573-0484. doi: 10.1007/s11227-015-1525-6. URL http://link.springer.com/10.1007/s11227-015-1525-6.

Machon Gregory and Ben Shneiderman. Shape Identification in Temporal Data Sets. In John Dill, Rae Earnshaw, David Kasik, John Vince, and Pak Chung Wong, editors, *Expanding the Frontiers of Visual Analytics and Visualization*, pages 305–321. Springer London, London, 2012. ISBN 978-1-4471-2803-8 978-1-4471-2804-5. doi: 10.1007/978-1-4471-2804-5_17. URL https://link.springer.com/10.1007/978-1-4471-2804-5_17.

M. Hao, M. Marwah, H. Janetzko, R. Sharma, D. A. Keim, U. Dayal, D. Patnaik, and N. Ramakrishnan. Visualizing frequent patterns in large multivariate time series. page 78680J, San Francisco Airport, California, USA, January 2011. doi: 10.1117/12.872169. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.872169.

Harry Hochheiser and Ben Shneiderman. Visual Queries for Finding Patterns in Time Series Data. 2001.

Harry Hochheiser and Ben Shneiderman. Visual Specification of Queries for Finding Patterns in Time-Series Data. 2002.

Dennis Janka, Felix Lenders, Shiyu Wang, Andrew Cohen, and Nuo Li. Detecting and locating patterns in time series using machine learning. *Control Engineering Practice*, 93:104169, December 2019. ISSN 0967-0661. doi: 10.1016/j.conengprac.2019.104169. URL https://www.sciencedirect.com/science/article/pii/S0967066119301601.

T. Kahveci and A.K. Singh. Optimizing similarity search for arbitrary length time series queries. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):418–433, April 2004. ISSN 1041-4347. doi: 10.1109/TKDE.2004.1269667. URL `http://ieeexplore.ieee.org/document/1269667/`.

Fritz Lekschas, Brant Peterson, Daniel Haehn, Eric Ma, Nils Gehlenborg, and Hanspeter Pfister. Peax : Interactive Visual Pattern Search in Sequential Data Using Unsupervised Deep Representation Learning. *Computer Graphics Forum*, 39(3):167–179, June 2020. ISSN 0167-7055, 1467-8659. doi: 10.1111/cgf.13971. URL `https://onlinelibrary.wiley.com/doi/10.1111/cgf.13971`.

Miro Mannino and Azza Abouzied. Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Montreal QC Canada, April 2018. ACM. ISBN 978-1-4503-5620-6. doi: 10.1145/3173574.3173962. URL `https://dl.acm.org/doi/10.1145/3173574.3173962`.

Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 1483–1492, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580111. doi: 10.1145/1357054.1357286. URL `https://doi.org/10.1145/1357054.1357286`.

Abdullah Mueen, Sheng Zhong, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance, August 2022. `http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html`.

Tamara Munzner. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, November 2009. ISSN 1077-2626. doi: 10.1109/TVCG.2009.111. URL `http://ieeexplore.ieee.org/document/5290695/`.

Panagiotis Papapetrou, Vassilis Athitsos, Michalis Potamias, George Kollios, and Dimitrios Gunopulos. Embedding-based subsequence matching in time-series databases. *ACM Transactions on Database Systems*, 36(3):

1–39, August 2011. ISSN 0362-5915, 1557-4644. doi: 10.1145/2000824.2000827. URL https://dl.acm.org/doi/10.1145/2000824.2000827.

Oliver Roesler. EEG Eye State. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C57G7J.

Maureen Stone. Choosing Colors for Data Visualization. January 2006.

J.J. van Wijk. The value of visualization. In *VIS 05. IEEE Visualization, 2005.*, pages 79–86, 2005. doi: 10.1109/VISUAL.2005.1532781.

Yuqing Wan, Xueyuan Gong, and Yain-Whar Si. Effect of segmentation on financial time series pattern matching. *Applied Soft Computing*, 38:346–359, January 2016. ISSN 1568-4946. doi: 10.1016/j.asoc.2015.10.012. URL https://www.sciencedirect.com/science/article/pii/S1568494615006341.

Martin Wattenberg. Sketching a graph to query a time-series database. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, page 381–382, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133405. doi: 10.1145/634067.634292. URL https://doi.org/10.1145/634067.634292.

Xiaomin Xu, Sheng Huang, Yaoliang Chen, Kevin Browny, Inge Halilovicy, and Wei Lu. Tsaaas: Time series analytics as a service on iot. In *2014 IEEE International Conference on Web Services*, pages 249–256, 2014. doi: 10.1109/ICWS.2014.45.

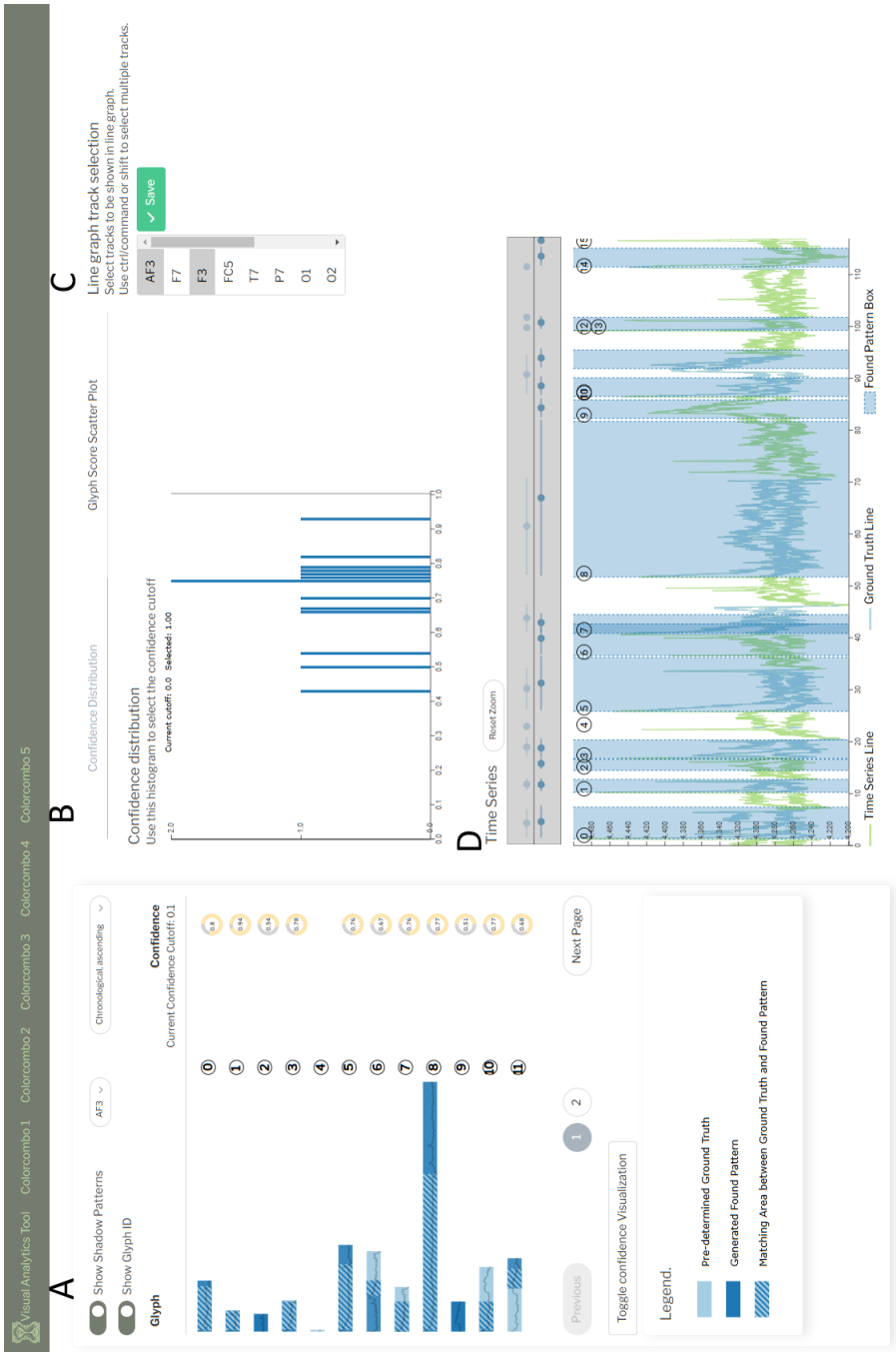
Li Yan, Nerissa Xu, Guozhong Li, Sourav S Bhowmick, Byron Choi, and Jianliang Xu. Sensor: data-driven construction of sketch-based visual query interfaces for time series data. *Proc. VLDB Endow.*, 15 (12):3650–3653, August 2022. ISSN 2150-8097. doi: 10.14778/3554821.3554866. URL https://doi.org/10.14778/3554821.3554866.

Yuncong Yu, Dylan Kruyff, Jiao Jiao, Tim Becker, and Michael Behrisch. PSEUDo: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–10, 2022. ISSN 1077-2626, 1941-0506, 2160-9306. doi: 10.1109/TVCG.2022.3209431. URL https://ieeexplore.ieee.org/document/9904933/.

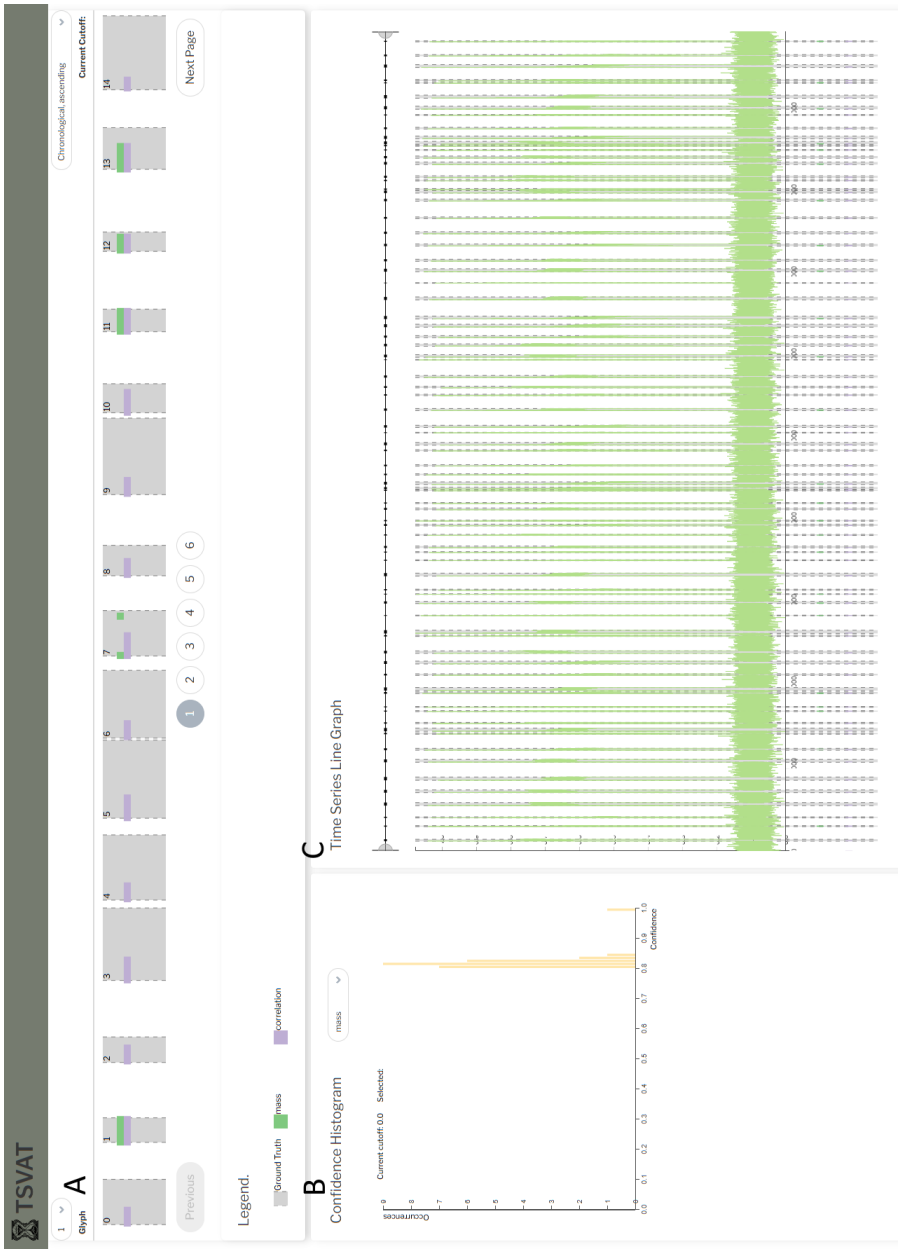
Yuncong Yu, Tim Becker, Le Minh Trinh, and Michael Behrisch. SAXRegEx: Multivariate time series pattern search with symbolic representation, regular expression, and query expansion. *Computers & Graphics*, 112:13–21,

May 2023. ISSN 00978493. doi: 10.1016/j.cag.2023.03.002. URL `https://linkinghub.elsevier.com/retrieve/pii/S0097849323000316`.

Appendix A Module Algorithm Exploration Overview



Appendix B Module Algorithm Comparison Overview



Appendix C Expert Study Questionnaire Answers

Algorithm Exploration Module

1. Did the algorithm exploration module meet your expectations?
   A: Yes.

2. What is, in your opinion, the most important element in the algorithm exploration module?
   A: It depends. The glyph chart is the most important, but still requires the line graph for analysis.

3. On a scale of 1 to 5, where 1 is really complicated and 5 is very easy, rate the difficulty of finding the features requested in the tasks within the algorithm exploration module.
   A: It was a bit complicated, because it is the first time I've seen it. After an explanation it was easier to understand. 4.

4. Was it easy to make sense of the interactions between different visualizations?
   A: Only thing that was harder to make sense of was the histogram, the other visualizations were very intuitive.

5. How easy was it to find the information you required in the tool?
   A: It was easy.

6. Do you think the algorithm exploration module provides additional opportunities for the evaluation of time series pattern search algorithms?
   A: I am not sure. Being able to save visualization results and visualization settings would be a nice opportunity.

7. Would you use the algorithm exploration module again in the future?
   A: Yes.

Algorithm Comparison Module

1. Did the algorithm comparison module meet your expectations?
   A: Yes.

2. What is, in your opinion, the most important element in the algorithm comparison module?
   A: It is different, compared to the other module. I think with this module filtering and sorting becomes more important.

3. On a scale of 1 to 5, where 1 is really complicated and 5 is very easy, rate the difficulty of finding the features requested in the tasks within the algorithm exploration module.
   A: Less features in this module, but still I would say 4.

4. Was it easy to make sense of the interactions between different visualizations?
   A: Yes, very cool.

5. How easy was it to find the information you required in the tool?
   A: Good enough. The histogram was again less easy.

6. Would you use the algorithm comparison module again in the future?
   A: Yes.