ChronoMantic: A Natural Language Interface for Interactive Time Series Pattern Query

Yangtian Liu
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, Zhejiang, China
yt-liu@zju.edu.cn

Shuhan Liu
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, Zhejiang, China
shliu@zju.edu.cn

Guobin Tu
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, Zhejiang, China
tuguobin@zju.edu.cn

Yunfan Zhou
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, Zhejiang, China
yf.zhou@zju.edu.cn

Xiwen Cai
Department of Digital Intelligence
China Mobile
Shenzhen, Guangdong, China
caixiwen@chinamobile.com

Di Weng*
School of Software Technology
Zhejiang University
Ningbo, Zhejiang, China
dweng@zju.edu.cn

Yingcai Wu
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, Zhejiang, China
ycwu@zju.edu.cn

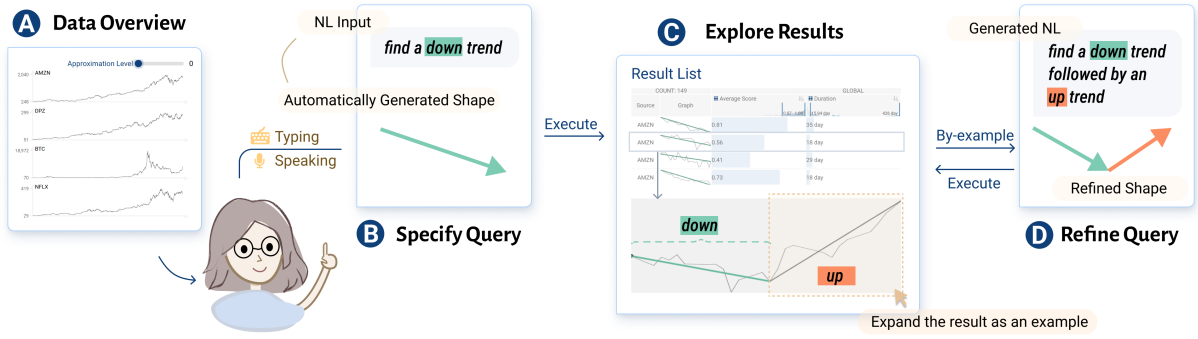

Figure 1: The workflow of *ChronoMantic*. (A) Users obtain a data overview to understand overall trends across approximation levels. (B) Users enter an NL query, and a glyph representation is generated in real time to show how *ChronoMantic* interprets the input. (C) After verifying correct parsing, users execute the query to explore results via filtering and sorting and inspect results in detail; they may refine the query to improve results or investigate observed patterns. (D) Users can refine the NL query by selecting an observed example to generate a new query without manual input, supporting iterative refinement.

Abstract

Finding time series patterns is an important problem in finance, healthcare, climatology, and manufacturing, where temporal data analysis informs critical decision-making processes. Existing approaches use examples, sketches, and natural language (NL) for time series pattern querying, but precisely formulating complex queries, inspecting query interpretations, and refining queries iteratively remain challenging. Nevertheless, NL is a promising modality for pattern specification, balancing expressiveness and preciseness while providing an intuitive means of articulating query intent. This paper introduces *ChronoMantic*, an NL-driven interactive interface for time series pattern querying. The tool provides glyph-based feedback on query interpretation and supports iterative querying through by-example refinement. We illustrate *ChronoMantic*'s workflow through two usage scenarios, assess its retrieval performance on a manually annotated benchmark, and evaluate its query effectiveness and usability via a two-stage user study.

CCS Concepts

• **Human-centered computing** → **Natural language interfaces**.

*Di Weng is the corresponding author.

UIST '26, Detroit, MI, USA

ACM ISBN 979-8-4007-2856-3/26/11
https://doi.org/10.1145/3830398.3830491

Keywords

Time series pattern querying, natural language interface

ACM Reference Format:

Yangtian Liu, Shuhan Liu, Guobin Tu, Yunfan Zhou, Xiwen Cai, Di Weng, and Yingcai Wu. 2026. ChronoMantic: A Natural Language Interface for Interactive Time Series Pattern Query. In *The 39th Annual ACM Symposium on User Interface Software and Technology (UIST '26), November 02–05, 2026, Detroit, MI, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3830398.3830491

1 Introduction

Time series data is a fundamental type of data across diverse domains, such as scientific research, financial analytics, and engineering systems [62], capturing temporal dynamics that are critical for understanding trends, making predictions, and detecting anomalies [31]. While the patterns within time series data provide valuable insights for various applications [17], it remains challenging for analysts to efficiently locate specific patterns of interest [16].

Prior work has explored leveraging different input modalities, including examples, sketches, and natural language (NL), to assist users in finding specific patterns. Query-by-example [9, 36] and by-sketch [16, 43] approaches allow users to specify the patterns of interest through representative examples and hand-drawn sketches, respectively. Query-by-NL approaches [5, 60] retrieve the desired patterns by translating the given NL instructions into underlying pattern specifications. However, we have identified the following three limitations in the existing approaches:

Limited preciseness of query formulation in the "intent → query" process. Due to the diverse and complex nature of queries, existing approaches struggle to allow users to precisely transform their intents to queries. Query-by-example approaches rely on existing pattern instances for retrieval, which can be challenging to find in a large amount of data. Query-by-sketch approaches tend to be sensitive to minor details of given sketches, such as unsteady strokes, making it difficult to effectively capture the user's intent [16]. While NL can flexibly express temporal features, a gap often exists between a user's query intent and their ability to construct a textual query that accurately describes it, particularly when targeting a complex pattern.

Limited transparency into query interpretation in the "query → result" process. Many existing interactive tools for time series pattern querying provide limited visibility into how user inputs are translated into executable query specifications. For example, since query-by-sketch tools like Qetch [43] lack explicit presentations of query specifications (e.g., trends), users cannot clearly anticipate how matching algorithms will interpret their hand-drawn sketches, complicating the verification of query conditions. Similarly, with query-by-NL approaches, when users provide complex NL instructions containing multiple conditions as in Figure 3 E1, they tend to struggle to verify whether their inputs are accurately translated into the intended query conditions.

Lack of support for iterative query refinement during exploration in the "result → query" process. Users' queries on time series data are often iterative, due to the exploratory nature of pattern discovery and the need to refine search criteria based on initial results. However, existing approaches do not sufficiently support such iterative querying. For instance, leveraging query-by-sketch approaches, users can draw a complex sketch to query a "*head-and-shoulders*" pattern like Figure 5 A2, but they have to manually redraw the entire pattern from scratch to query an inverted one like Figure 5 B2. Alternatively, query-by-NL approaches enable iterative refinement by allowing users to directly edit textual queries. However, users must manually translate visual patterns of interest discovered in the results into descriptive language to investigate these discoveries further.

Nevertheless, NL offers a balance of expressiveness and preciseness [29], capable of capturing both high-level intents (e.g., "*head-and-shoulders*") and nuanced constraints (e.g., "*with a one-day duration*"). Recent advances in large language models (LLMs) further enhance interpretation of such queries, making NL a promising modality for specifying time series patterns. To address the mentioned limitations, we developed *ChronoMantic*[1], an NL-driven interface for time series pattern querying, enhanced with visual query-interpretation feedback and interactive query refinement.

For the first and third limitations, *ChronoMantic* combines direct NL input with by-example refinement to support query construction and iteration. Users can either directly enter NL instructions or select existing time series segments as auxiliary examples to formulate queries for target patterns. For the second limitation, *ChronoMantic* parses NL into query conditions in real time with explanatory visual glyphs. Meanwhile, it highlights the corresponding text in the NL query to help users verify whether the interpreted conditions align with their intent. We illustrate *ChronoMantic*'s workflow through two usage scenarios and assess its retrieval performance on our manually annotated time series pattern dataset. Furthermore, to evaluate *ChronoMantic* in practical use, we conducted a two-stage user study that compared the query effectiveness of *ChronoMantic* with *Qetch* in the first stage and assessed the usability of *ChronoMantic* through an exploratory query simulation. *ChronoMantic* achieved higher query effectiveness than *Qetch* across complexity levels; participants also valued the glyph-based interpretation feedback and by-example refinement.

In summary, our contributions are as follows:

- *ChronoMantic*, a novel NL-driven interactive interface for time series pattern querying that provides intuitive interpretations of NL input via visual glyph feedback and enhances iterative querying through by-example refinement.
- A two-stage user study evaluating *ChronoMantic*'s effectiveness and usability in practical time series pattern querying.

2 Related Work

We review research on time series query specification, matching, and natural language interfaces for data analysis. Query specification concerns formulation methods via diverse modalities, while matching involves algorithms for time series retrieval.

2.1 Time Series Query Specification

Query specification is the initial step in the query processes, enabling users to define patterns through various modalities.

Constraint-based approaches use thresholds [17] or timeboxes (value and time ranges) [9, 23, 73] for filtering. Syntactic methods

[1]The source code is available at https://github.com/SunskyLau/ChronoMantic.

employ custom syntax or regular expressions [1, 60, 65], while recent tools support multivariate relational constraints [39] or graphical constraint lines [55].

Query-by-example systems [9, 65] retrieve patterns similar to provided inputs. Some interactive interfaces [36, 72] refine results via relevance feedback. However, their utility is constrained when users cannot locate initial representative examples.

Sketch-based methods [39, 43] offer intuitive specification via three main forms: overlaying inputs directly on data [13, 16, 69]; free-form canvases with annotations (e.g., scales) [24, 43]; and simplified linear or grid-based inputs [32, 39, 47, 54] that prioritize main trends over details. Despite their intuitiveness, sketching complex temporal features remains challenging.

Natural language (NL) approaches align with how people naturally express their intents. Tools like Qute [29], ShapeSearch [60], and SemanticSearch [65] support queries via restricted vocabularies, while Bamford et al. [5] leverage NL for high-level features. Furthermore, SlopeSeeker [6] and Bromley et al. [7] focus on quantifying semantic labels for visual trends to enhance NL-based search and analysis. However, existing methods typically suffer from restricted expressiveness and flexibility.

ChronoMantic combines NL input with by-example refinement for complex, iterative querying and uses glyph-based feedback to externalize semantic conditions defined by a structured grammar.

2.2 Time Series Query Matching

Matching algorithms are fundamental for retrieving relevant data. Metrics like Euclidean distance [14] handle equal-length comparisons, while Dynamic Time Warping (DTW) [46] and Fast-DTW [56] accommodate temporal distortions via nonlinear alignment. Symbolic Aggregate Approximation (SAX) [38, 54] reduces dimensionality by transforming series into strings. Feature-based methods, such as Qute [29], map queries to computational feature vocabularies. Machine learning approaches further expand capabilities, utilizing LSTM networks for sketch queries [16] or deep encoders for multimodal inputs [5]. Specialized databases (e.g., InfluxDB [30], TimescaleDB [63], KDB+ [35], Timestream [2]) optimize management via indexing and downsampling. Finally, linear fitting methods [1, 47, 60] facilitate trend-based querying by identifying fundamental trends (increasing, flat, decreasing) and their combinations.

ChronoMantic employs a bottom-up merging approach with piecewise linear fitting to segment time series at multiple approximation levels. This enables efficient feature extraction and pattern matching across diverse time scales.

2.3 NL Interfaces for Data Analysis

Natural Language Interfaces (NLIs) facilitate data analysis across visualization [12, 25, 40, 45, 59, 61, 66, 67], transformation [28], and management [3, 34, 48], enabling non-experts to interact conversationally. Implementation typically comprises NL understanding and command execution. (1) For understanding, traditional rule-based methods [18, 27, 49, 58, 71] rely on syntactic parsing [42], offering reliability for structured queries but struggling with ambiguity. Deep learning approaches [33, 41, 68] improve sophistication but demand substantial training data. Recently, LLMs [4, 15, 50] have enabled robust handling of complex queries and domain generalization [28, 64]. (2) Command execution transforms intent into actionable operations via intermediate representations [27, 28, 41, 49]. Targets include DSLs (e.g., SQL), visualization grammars like Vega-Lite [57], or system-specific commands, depending on the domain.

Following this pipeline, *ChronoMantic* uses an LLM to translate NL queries into structured conditions that guide the retrieval of relevant time series patterns.

3 Formative Interview

This section reports a formative interview[2] with two goals: to understand analysts' workflows for querying time series patterns and the limitations of current interactive tools, and to derive design requirements for a more effective system. To prompt discussion, we demonstrated tools that support query specification through different modalities and then conducted semi-structured interviews. Interview findings informed the design of *ChronoMantic* to streamline manual workflows and address limitations in existing tools.

3.1 Setup

3.1.1 Participants. Six time series analysts (P1–P6) were recruited via verbal invitations and social media. They had 1–3 years of professional experience across domains including stock analysis, transportation flow analysis, and photovoltaic power generation. All participants consented to audio recording.

3.1.2 Procedure and Materials. After a brief introduction, we presented three tools: Qetch [43], TimeSearcher2 [9], and an NL-driven prototype with basic query functionalities. The prototype included a table view for loaded time series data, an NL query box, and a result panel showing basic pattern information (e.g., duration, start time, and end time). We built the prototype because no suitable open-source NL-based time series query tools were available for demonstration. Participants were encouraged to ask questions during the demos. We then conducted semi-structured interviews to understand workflows for time series pattern queries and collect feedback on limitations. The interview protocol and a prototype screenshot are provided in the supplementary material.

3.2 Findings

After the interviews, the first author analyzed notes and voice recordings to organize feedback. The co-authors then discussed and synthesized the feedback in weekly meetings until reaching consensus on the key findings below.

3.2.1 The Workflow of Time Series Pattern Querying. A typical workflow starts with a **preliminary data overview** to observe overall trends. Analysts then either focus on time periods of interest to form query intents or conduct **initial queries** based on objectives. For straightforward tasks, three participants (P1, P5, P6) mainly rely on visual inspection; two (P2, P3) use tools to specify constraints for automated queries; and one (P4) sometimes uses advanced scripting. They **explore the results** to find intent-aligned matches and enter an **iterative query process** to refine intents, guided by result quality and surrounding segments (i.e., the immediately preceding and following segments).

[2]The interview has been approved by State Key Lab of CAD&CG, Zhejiang University.

3.2.2 Limitations of Existing Approaches. We identified three limitations from user feedback:

Difficulties in precisely authoring queries. Participants valued these methods but struggled to express intent precisely. For sketching, inaccuracies could affect results, as P1 and P3 stated: "*Casual sketching might introduce elements that are difficult for the system to interpret.*" For NL, the challenge is structuring language to specify patterns; P4 explained, "*I find it challenging to write multiple conditions in NL.*" By-example querying was intuitive but restrictive due to reliance on single examples; P3 and P5 noted: "*This method is overly dependent on existing examples; if the example doesn't contain the pattern I need, specifying queries becomes impossible.*"

Uncertainty about how the query is processed. Participants worried about how tools interpreted queries, especially for unexpected or empty results, echoing known challenges in interactive data systems [19, 21]. P6 noted, "*Even though the results generally meet expectations, I still worry about the comprehensiveness of the results since I don't know how the system interprets my NL query.*" P1 said, "*When the results don't meet my expectations, I'm unsure which specific part of the query the system failed to understand correctly.*"

Inability to iteratively refine queries. Sketching often requires redrawing to refine queries; as P5 said, "*This can be particularly cumbersome, as even minor adjustments require starting from scratch.*" NL refinement increases cognitive burden when adding conditions; for example, P2 stated, "*I have to organize my expressions to add new query conditions based on my original NL.*" P3 and P6 further noted that by-example methods do not allow users to specify which example features should guide refinements.

3.3 Design Requirements

Based on our findings, we derived four design requirements **(R1–R4)** to support the observed workflow and mitigate limitations of existing approaches.

R1: Provide a scalable overview of time series data. Since querying often starts with a "preliminary data overview" to observe trends, the tool should support exploration across time ranges and at multiple levels of granularity within a range.

R2: Offer intuitive and verifiable query interpretations. To reduce "uncertainty about how the query is processed", the tool should make interpretation transparent by showing how inputs are parsed and how their parts map to query conditions. It should provide controls to refine parsing and correct misinterpretations.

R3: Support efficient exploration of query results. To support "exploring the results" for intent-aligned matches, the tool should enable rapid retrieval and support navigating and filtering across large result sets, helping users identify relevant patterns.

R4: Facilitate iterative query refinement. Given limited support for iterative querying, the tool should enable progressive query modification as intents evolve or results motivate changes, reducing the burden of restarting and supporting complex intents.

4 ChronoMantic

In this section, we first provide an overview of *ChronoMantic*'s user interface and backend components (Section 4.1). We then demonstrate its core functionalities and underlying design via a usage scenario covering the complete workflow (Figure 1): data overview (Section 4.2), query initialization (Section 4.3), result exploration (Section 4.4), and query refinement (Section 4.5).

4.1 System Overview

ChronoMantic's user interface (Figure 3) comprises four components: a profile panel (C) for multi-scale trend inspection, a query panel (A) for direct NL input and visual interpretation feedback, a result panel (B) for exploring matches, and a detail panel (D) for inspecting results in detail and supporting by-example query refinement. Four backend components drive the system: a *data preprocessor* for multi-level approximation and feature extraction, a *query parser* translating NL into structured conditions, a *query executor* retrieving matching subsequences, and a *query generator* synthesizing NL from examples.

In terms of implementation, *ChronoMantic* is built as a web application using React [53] and Flask [51]. *ChronoMantic* uses a replaceable LLM API backend; the prototype used in the study was powered by the Qwen3-8B model [70] (temperature 0, maximum context length 1240 tokens). The implementation details are provided in the supplementary material.

4.2 Data Overview

James, a utility analyst, imported residential power data to investigate consumption patterns. He examined it at various approximation levels in the profile panel (Figure 2 A). After observing stable periods (Figure 2 A1), he wanted to find more such patterns.

To provide scalable data overviews (**R1**), *ChronoMantic*'s profile panel visualizes time series via line charts at adjustable approximation levels, facilitating efficient initial exploration. For granular analysis, the detail panel enables precise inspection of specific ranges through a resizable sliding window.

We employ a bottom-up merging approach with piecewise linear fitting for preprocessing (see supplementary material). Starting with $n-1$ initial segments $S = \{s_1, \ldots, s_{n-1}\}$ for a time series of size n, we iteratively merge adjacent pairs that minimize the increase in total squared error. This metric prioritizes geometric fidelity for semantic retrieval. The preprocessor stores approximations whenever the segment count is halved and extracts features at each level to support subsequent retrieval (Section 4.4).

4.3 Query Initialization

With a preliminary query intent, James inputs a query: "*Find patterns that exhibit a stable trend in global active power*" (Figure 2 B1). *ChronoMantic* parses this in real time, generating text highlights and an explanatory visual glyph (Figure 2 B2).

To facilitate verification of query interpretation (**R2**), the system provides immediate visual feedback via a glyph and text highlights, grounded in a custom query specification grammar (see supplementary material). The glyph visualizes parsed conditions: basic trends (uptrend/flat/downtrend, Figure 4 A), temporal boundaries (Figure 4 B), value scopes (Figure 4 C), and attributes like slope and relative slope (percentage of the steepest slope, Figure 4 D). Complex relations between trends (duration, slope, value) are denoted by comparison symbols ($<$, $>$, $=$, $\approx$) rather than direct geometric

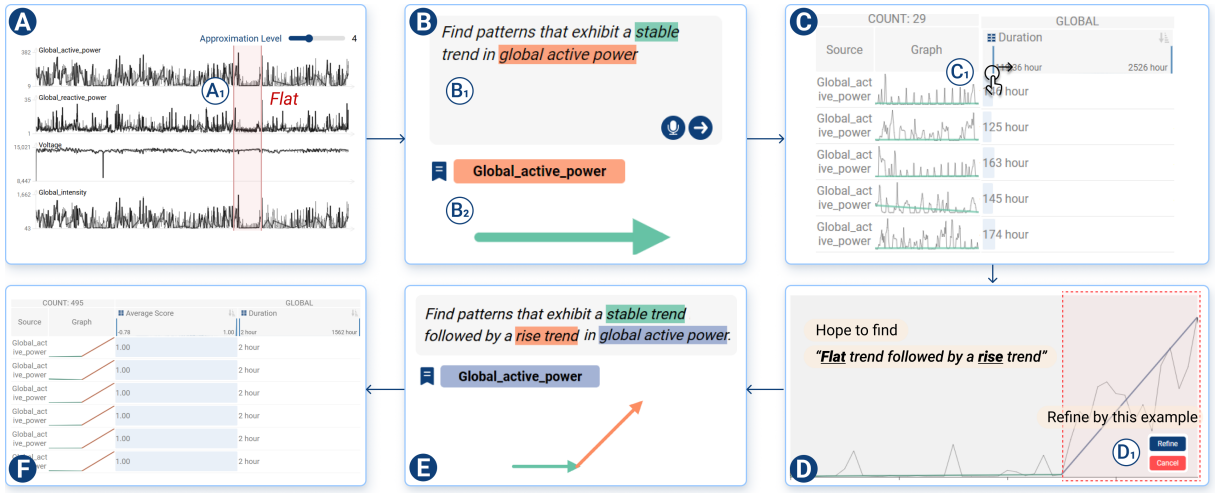

Figure 2: A usage scenario showing how an analyst investigates simple patterns in electricity consumption data with *ChronoMantic*. (A) Dataset overview. (B) Query initialization. (C) Result exploration. (D) By-example query refinement based on an examined result. (E) Generated NL query. (F) Updated results.

Figure 3: *ChronoMantic*'s interface and a usage scenario illustrating exploratory querying on an atmospheric pressure dataset. (A) Users enter NL queries; glyphs are generated in real time. (B) Users explore results via filtering and sorting. (C) The profile panel provides a dataset overview across approximation levels. (D) Users inspect results, select segments as examples, and optionally add conditions to generate refined NL queries. (E) The generated refined query. (F) Results from the refined query.

encoding (Figure 4 E-G). This symbolic approach avoids visual conflicts from geometric coupling; for instance, extending a segment's length to represent duration would unintentionally alter its vertical magnitude if slope is preserved.

Consistent color encoding links highlighted text to glyph elements. Users can click highlights to toggle specific conditions, exploring their influence on results. For parsing errors, users can right-click the glyph to open a specification panel (see supplementary material). As a fallback repair mechanism, the panel allows manual adjustment of parsed conditions and updates the glyph in real time. We employ an LLM to parse NL using a few-shot prompting strategy [52] that incorporates task descriptions, grammar definitions, and illustrative examples.

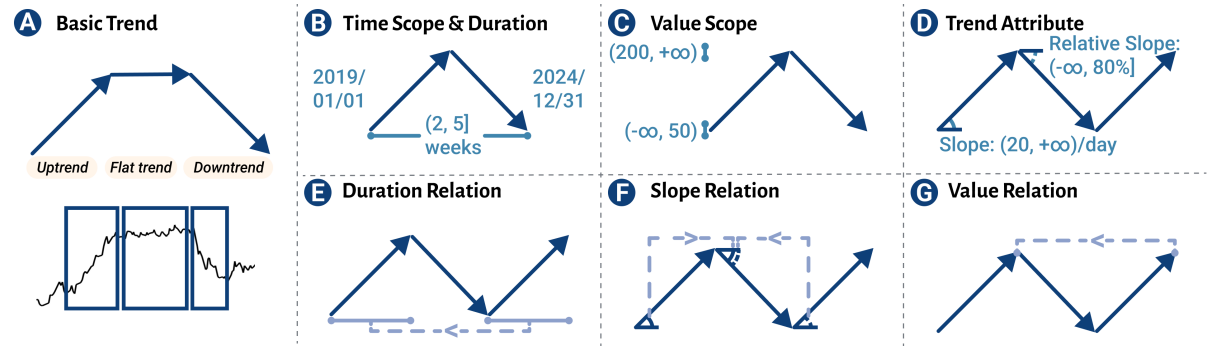

Figure 4: Query conditions and glyph design in *ChronoMantic*. By specifying attributes for trends (A, B, C, D) and relations between different trends (E, F, G), *ChronoMantic* can support various temporal pattern queries (e.g., the left trend in E has a shorter duration than the right trend). The explanatory glyph helps users understand how the NL inputs are parsed.

4.4 Result Exploration

James executes the verified query, populating the result panel with matches (Figure 2 C). To remove irrelevant short-duration patterns, he filters them by brushing the duration column header (Figure 2 C1), isolating results suitable for detailed exploration.

To support efficient exploration (**R3**), *ChronoMantic* retrieves results via the query executor and presents them in a multi-attribute result panel, while the detail panel reveals surrounding trends.

Upon execution, the executor receives structured conditions (Section 4.3) and scans all pre-computed segments (Section 4.2). It verifies contiguous subsequences by directly comparing cached features (e.g., slope, duration) and relational constraints against query conditions. This single-pass evaluation operates across all approximation levels; since the segment count halves at each level, the total search space sums to $O(n)$ (where n is the data point count), ensuring linear-time retrieval.

Inspired by LineUp [20], the result panel lists matches as rows with columns for IDs, color-coded thumbnails matching the glyph, and extracted features. Headers incorporate area charts for data distribution and support filtering and sorting. We calculate average segment R^2 as a linearity score to aid pattern selection. The view defaults to basic features (e.g., score, duration) but supports expansion via a "+" icon to reveal additional pre-calculated metrics.

Selecting a row magnifies the pattern in the detail panel, preserving the glyph's color scheme. The view also shows four preceding and four subsequent segments, enabling users to examine each match in context and formulate new query intents. Hover interactions enable precise inspection of data values.

4.5 Query Refinement

After inspecting several results in detail, James identifies a pattern of interest comprising a stable trend followed by a rise and expands it as an example (Figure 2 D). He clicks the "Refine" button (Figure 2 D1), prompting the system to update the NL query to "*Find patterns that exhibit a stable trend followed by a rise trend…*" and adjust the glyph (Figure 2 E). Finally, he executes this query to retrieve refined results (Figure 2 F).

To support iterative querying (**R4**), the query generator synthesizes new NL queries from user-selected examples. A retrieved result in the detail panel serves as a customizable template, expandable via adjacent segments. Clicking a segment activates a condition panel to specify features (e.g., slope, duration), with trend category enabled by default. Users can further brush segment groups to define duration constraints or use Shift+click to specify relational conditions. Clicking "Refine" generates the corresponding NL. Alternatively, users can "Author" queries from scratch by selecting arbitrary segments and specifying conditions similarly, without relying on prior results.

Similar to the query parser (Section 4.3), we use few-shot prompting for the query generator. The prompt includes the task description, dataset metadata, the original NL query and its corresponding parsed conditions, example attributes, grammar definitions, and illustrative examples. We ask the LLM to simultaneously output the new NL query and its corresponding structured conditions.

5 Usage Scenarios

We demonstrate *ChronoMantic*'s versatility through two scenarios. The first employs typed input, while the second features by-example refinement to facilitate exploratory querying.

5.1 Purposeful Querying in a Financial Dataset

Sarah, a finance student, uses *ChronoMantic* to investigate common stock patterns in real-world datasets. She begins by inputting "*Find head-and-shoulders patterns*" (Figure 5 A1). *ChronoMantic* instantly generates a glyph featuring three peaks, with the middle one highest (Figure 5 A2). After confirming that the glyph accurately reflects the technical definition, Sarah executes the query. She then inspects the returned thumbnails and details (Figure 5 A3), using the shared color encoding to compare each match with the interpreted conditions. Observing the correspondence between the returned results and the glyph reinforces Sarah's understanding that retrieval is grounded in the parsed conditions.

Next, to find "*inverted head-and-shoulders*" patterns, she simply adds the word "*inverted*" to her previous query (Figure 5 B1). The

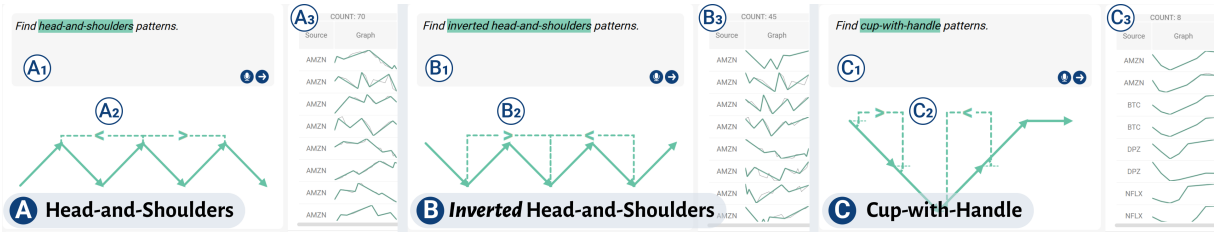

Figure 5: A usage scenario demonstrating a purposeful query process with *ChronoMantic* on a financial stock dataset. (A) Query for "head-and-shoulders" patterns. (B) Query for "inverted head-and-shoulders" patterns by adding "inverted". (C) Query for "cup-with-handle" patterns by replacing "inverted head-and-shoulders" with "cup-with-handle".

glyph updates immediately to an inverted version (Figure 5 B2). After verifying this visual feedback, she executes the query, retrieving a set of matching inverted patterns (Figure 5 B3).

Finally, she tests for "*cup-with-handle*" patterns (Figure 5 C1). The system again presents its interpretation through a glyph (Figure 5 C2), which she verifies before inspecting the returned matches (Figure 5 C3). This session demonstrates how NL queries and glyph-based interpretation work together during pattern exploration.

5.2 Exploratory Querying in an Atmospheric Pressure Dataset

Bob, a meteorologist, investigates atmospheric pressure data using *ChronoMantic*. He starts with a basic query, "*Find patterns that exhibit a peak*" (Figure 3 A1), and verifies the generated "up then down" glyph (Figure 3 A2). After execution, he filters out excessive durations via the result panel handle (Figure 3 B) to focus on relevant scales (Figure 3 D).

During inspection, Bob discovers an intriguing anomaly: a peak followed by a second peak of equal height (Figure 3 D1). Eager to find similar rare instances without manual reformulation, he leverages by-example refinement. He expands the pattern to include the second peak, uses Shift+click to define an equal-height relation between the rising trends (Figure 3 D2), and brushes the example to constrain the duration (Figure 3 D3).

Clicking "Refine" prompts the system to generate a precise query: "*Find periods exhibiting two consecutive peaks of approximately equal height…*" (Figure 3 E1) with a corresponding complex glyph (Figure 3 E2). Validating that this aligns with his intent, Bob executes the query. The retrieved patterns reveal a concentration near global pressure peaks, suggesting a potential signal for pressure trend reversals (Figure 3 F). This workflow highlights *ChronoMantic*'s capacity to support iterative, insight-driven exploration.

6 Technical Evaluation

To assess *ChronoMantic*'s retrieval performance, we conducted a comparative evaluation on a manually annotated time series pattern dataset, using *Qetch* [43] as a representative open-source baseline.

6.1 Baseline

We surveyed time series query tools using three criteria: (a) pattern-based subseries retrieval, (b) support for a range of patterns comparable to ours, and (c) an open-source implementation. We considered NL-based tools (*ShapeSearch* [60], *Qute* [29], *SlopeSeeker* [6]) and sketch-based tools (*Qetch* [43], Fan et al. [16]) (a detailed comparison is provided in the supplementary material). We chose *Qetch* as it best satisfies these criteria and its interactive retrieval objective aligns with *ChronoMantic*.

6.2 Dataset Construction

As no suitable benchmark exists, we created one by manually annotating patterns in a financial portfolio dataset[3]. Following *ShapeSearch*'s principle that complex patterns typically involve no more than six trend lines [60], we defined 14 patterns spanning 3–6 trend lines (e.g., *plateau*, *basin*, *head-and-shoulders*). The first and third authors annotated all patterns independently using a custom tool that shows a line chart and supports brush selection under multiple zoom scales. Because interval discovery was open-ended, we assessed overlap-based agreement rather than categorical kappa: one-to-one matches with IoU > 0.75 were considered consistent, yielding $2M/(N_1+N_2) = 0.677$, where M is the number of consistent pairs and N_1 and N_2 are the annotators' totals. We de-duplicated matched intervals and jointly reviewed unmatched intervals to produce the final 240 annotations across 14 patterns; the dataset is included in the supplementary material.

6.3 Experimental Procedure

We designed the procedure to account for differences in input modalities and parameters. For *ChronoMantic*, we ran one trial per pattern because identical NL inputs produced consistent results. For *Qetch*, to handle sketch variability, three authors each created one sketch per pattern, yielding three sketches per pattern. We also evaluated five distance thresholds (5, 7.5, 10, 25, 40), as *Qetch* defines good matches as distance < 5, fair matches as 5–40, and poor matches as > 40. For each threshold, we tested three sketches for each pattern and retained the one with the highest F1 (Section 6.4). We then selected the threshold with the highest overall F1 as *Qetch*'s final configuration. All NL and sketch inputs are provided in the supplementary material.

6.4 Evaluation Metrics

Following Holz et al. [24], we used intersection-based matching to compute micro-precision, micro-recall, and micro-F1. For each pattern, we computed an IoU matrix between ground-truth annotations and query results, greedily matched the maximum-IoU pair

[3]https://www.kaggle.com/datasets/hershyandrew/amzn-dpz-btc-ntfx-adjusted-may-2013may2019

as a true positive, removed the pair, and repeated until no IoU exceeded 0.75. Unmatched results were false positives and unmatched annotations were false negatives. We then computed $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, and $F1 = \frac{2\times P\times R}{P+R}$, micro-averaged across patterns.

6.5 Result Analysis

Results (the complete table is in the supplementary material) show that *ChronoMantic* outperforms *Qetch*. *ChronoMantic* achieved $F1 = 0.235$ ($P = 0.148$, $R = 0.562$), while the best *Qetch* configuration ($distance\_threshold = 5$) achieved $F1 = 0.052$ ($P = 0.030$, $R = 0.212$). This represents a greater than four-fold improvement in F1. We attribute this gap to underlying retrieval mechanisms: *Qetch* relies on shape/area distances without semantic factors, retrieving metrically similar but semantically incorrect matches (lower precision) and missing perceptually relevant variants (lower recall). *ChronoMantic* maps semantic descriptions into structured conditions and matches segment sequences, better aligning with the benchmark's perceptual criteria and retrieving relevant variations. Despite the relative improvement, absolute precision remains low for both systems; sorting, filtering, thumbnails, and detail views support post-hoc inspection of returned matches and help users identify the most relevant results.

To better understand the low precision, we manually reviewed a sample of false positives and observed two recurring sources. First, local noise or distortions sometimes appeared to disrupt the segmentation process, producing boundaries or levels of granularity that did not align with how human observers perceived the patterns. Second, the thresholds assigned by the system to parsed conditions on attributes such as slope and duration sometimes appeared more permissive than human perceptual criteria, thereby admitting results that formally satisfied the constraints but did not visually resemble clear instances of the target patterns.

We also conducted a post-hoc sensitivity analysis of the mean fitted-segment R^2 threshold, which excludes results below the specified cutoff. Relative to the unfiltered baseline reported above, performance changed little at $R^2 \geq 0.40$ ($P = 0.149$, $R = 0.554$, $F1 = 0.235$) and reached the highest $F1$ at $R^2 \geq 0.60$ ($P = 0.157$, $R = 0.525$, $F1 = 0.242$). At $R^2 \geq 0.85$, precision increased modestly but recall declined substantially ($P = 0.178$, $R = 0.283$, $F1 = 0.219$), indicating a threshold-dependent precision–recall trade-off.

To probe parser robustness to rephrasing with a recent high-capability LLM, we configured *ChronoMantic*'s parser backend to use Gemini 3.5 Flash. Across three paraphrases for each of eight query intents, the parser correctly mapped all 24 queries to their intended structured conditions. The benchmark comparison aims to evaluate retrieval performance under the implemented systems, rather than the isolated effect of any single query modality.

7 User Study

In this section, we report a two-stage user study[4] evaluating *ChronoMantic*: Stage 1 assessed query effectiveness in practical use via a comparative experiment with *Qetch*, and Stage 2 assessed usability through a simulated exploratory querying task. Further details are provided in the supplementary material.

7.1 Setup

7.1.1 Participants and Apparatus. We recruited 16 participants (S1–S16; 10 males, 6 females) via social media (18–28 years; $M = 22.56$, $SD = 2.87$). Their familiarity with time series data was assessed on a 7-point Likert scale ($M = 4.63$, $SD = 1.45$). Participants had diverse backgrounds (computer science, medicine, management, psychology, accounting, and control science). Each received $8.65 upon completing the study. Sessions were conducted on a Windows 11 workstation with a 27-inch monitor (3840×2160). We recorded screens and generated transcripts using meeting software.

7.1.2 Stage 1: Fixed Pattern Querying. In this stage, we ran a within-subjects comparative experiment to evaluate *ChronoMantic*'s effectiveness in practical querying. We consistently selected *Qetch* [43] as the baseline, with the rationale discussed in Section 6.1. As both tools are intended to be accessible to a broad range of users, we followed *Qetch*'s original setup and used mouse input, which does not require specialized pen or tablet hardware.

Dataset. We used the dataset[5] (daily ALK opening prices, 1997-08-22 to 2002-08-22) provided in the *Qetch* repository.

Tasks. We designed four tasks with increasing complexity in combinations of trend lines (L1–L4), following *ShapeSearch*'s principle [60] that even complex patterns typically involve no more than six trend lines. Each task presented three example images with graphical and textual annotations; the text matched the images in granularity to ensure a fair comparison between *ChronoMantic* and *Qetch*. For each task, participants used both tools to find matches within 3 minutes and could refine queries during the search. After each search, they evaluated results by browsing the result panels. We ensured each pattern could be specified in both tools. To reduce sequence and learning effects, we counterbalanced tool order and task patterns across participants.

Procedure. This stage comprised four phases: *(1) Tutorial.* We introduced the experiment, demonstrated both tools, and explained the glyph to convey the query grammar's expressive space; questions were answered throughout. *(2) Warm-up.* Participants practiced with both tools to familiarize themselves with key functions. *(3) Pattern query tasks.* Participants completed four tasks (L1–L4) with each tool; we recorded completion time and collected satisfaction ratings on a 7-point Likert scale (1–strongly unsatisfied, 7–strongly satisfied) after each task. *(4) Semi-structured interview.* After all tasks, participants completed six NASA-TLX questions [22] using a 7-point scale and participated in a semi-structured interview on workload and tool preferences across complexity levels. The stage lasted ~40 minutes.

7.1.3 Stage 2: Exploratory Pattern Query Simulation. Stage 2 assessed *ChronoMantic*'s usability in exploratory scenarios, complementing Stage 1's focus on fixed pattern querying effectiveness. We designed tasks to simulate exploratory pattern querying, emphasizing iterative querying and exploration. Participants used only *ChronoMantic* because this stage was designed to simulate its exploratory querying workflow rather than compare systems.

Dataset. We used the portfolio dataset described in Section 6.2.

[4] The user study has been approved by State Key Lab of CAD&CG, Zhejiang University.

[5] https://github.com/huda-lab/qetch/blob/master/Datasets

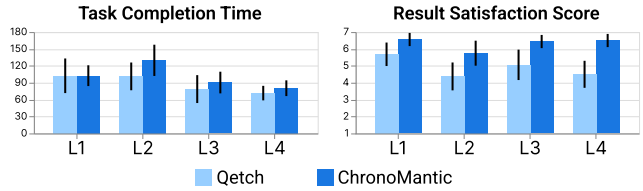

Figure 6: Quantitative results of Stage 1 tasks. We recorded the task completion time (in seconds) and result satisfaction ratings for both *Qetch* and *ChronoMantic* across four task levels (L1–L4).

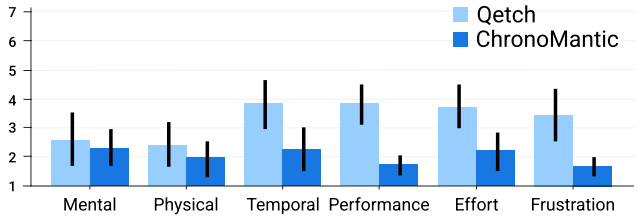

Figure 7: The perceived workload of participants using *Qetch* and *ChronoMantic*.

Tasks. Tasks simulated the workflow in Section 3.2.1 (dataset overview, query initialization, result exploration, and query refinement) and required use of all major *ChronoMantic* functions.

Procedure. This stage had four phases: *(1) Tutorial.* We demonstrated *ChronoMantic*'s by-example features; participants could ask questions at any time. *(2) Warm-up.* Participants practiced the by-example functionality to ensure understanding. *(3) Exploratory pattern query simulation tasks.* Participants had 10 minutes to complete the simulation tasks. *(4) Semi-structured interview.* Participants completed a SUS questionnaire [8] (5-point Likert scale), followed by a semi-structured interview based on their SUS responses. This stage lasted ~20 minutes.

7.2 Quantitative Analysis of Results

Figure 6 (right) shows satisfaction scores for *ChronoMantic* and *Qetch*; all tasks were completed within the time limit. Some participants finished certain tasks much faster than average but gave low scores. This may be because when dissatisfied with initial results, they stopped further exploration and refinement; thus, we did not analyze completion time and focused on scores. Because the 7-point Likert satisfaction data violated normality or homogeneity of variance, we used the Scheirer–Ray–Hare test as a rank-based nonparametric alternative to two-way ANOVA. For satisfaction, the tool×complexity interaction was not significant ($p = 0.561$), while the main effects of tool ($p < 0.001$) and complexity ($p < 0.05$) were significant; rank η^2 values were 0.233, 0.070, and 0.016 for tool, complexity, and their interaction, respectively. Post-hoc comparisons used two-sided within-participant paired Wilcoxon signed-rank tests with Holm correction and showed higher scores for *ChronoMantic* at all complexity levels (L1: $p_{\text{adj}} = 0.022$, L2: $p_{\text{adj}} = 0.024$, L3: $p_{\text{adj}} = 0.024$, L4: $p_{\text{adj}} = 0.006$), with paired rank-biserial correlations ranging from $r_{\text{rb}} = 0.736$ to 1.000. Overall, *ChronoMantic* achieved higher satisfaction than *Qetch*.

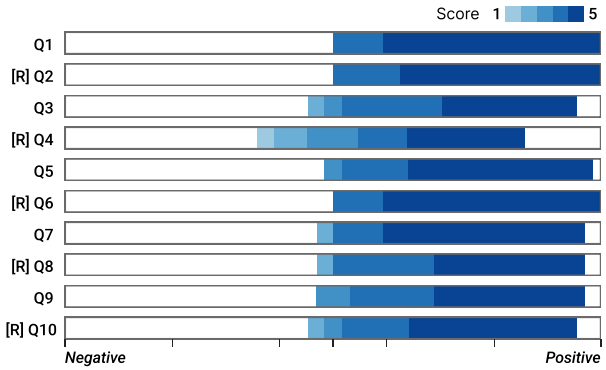

Figure 8: System usability results from Stage 2 using a 5-point SUS questionnaire [8]. For better comparison, the scores for negatively worded items (Q2, Q4, Q6, Q8, Q10) are presented in inverted form.

Figure 7 shows NASA-TLX results: participants perceived lower workload with *ChronoMantic* than *Qetch* across all six aspects (mental demand, physical demand, temporal demand, performance, effort, and frustration), suggesting reduced workload in pattern querying. *ChronoMantic* achieved an average SUS score of 87.66 (Figure 8), exceeding the 84.1 threshold for an A^+ rating in Jeff Sauro's survey [44]. Following the SUS factor-structure study [37], the usability score was 90.04 and the learnability score was 78.13.

7.3 Qualitative Analysis of Results

We conducted content analysis of the interviews. The first and third authors independently open-coded the transcripts and reconciled discrepancies to consolidate comments into themes. The first author synthesized the codes into the key findings below.

7.3.1 Result Quality and Presentation. All participants (16/16) reported positive feedback on *ChronoMantic*'s query results, describing them as "*accurate*" (S1, S3, S4, S5, S6, S8–S10), "*align well with expectations*" (S7, S9, S11, S13, S14), "*good*" or "*better*" (S2, S3, S7, S10, S11, S12, S13, S15, S16), and "*comprehensive*" or "*abundant*" (S14, S15). As S3 noted, "*ChronoMantic demonstrates impressive intelligence. Its LLM parses even my most abstract natural language with surprising accuracy, outperforming my expectations.*" Most participants (11/16) also stated that *Qetch* could not adequately capture sketch intent; S15 stated, "*I noticed Qetch's results don't match my sketched intents — I drew a steep curve, but it returned gradual patterns.*" Four (S4, S9, S14, S16) further highlighted *ChronoMantic*'s result presentation: S4 and S9 valued its comprehensive information, while S14 and S16 appreciated its sorting and filtering functions.

7.3.2 Sketch vs. NL. Most participants (14/16) commented positively on sketching, describing it as "*convenient*" (S5, S6, S7, S9, S10, S15, S16), "*more intuitive*" (S13), "*faster*" (S2, S13), "*low-effort*" (S11, S14), and "*easier to use*" (S3–S5, S8). However, five (S1, S6, S9–S11) reported mouse-based sketching issues with convenience and

controllability; as S6 said, "*When I first tried it, I thought sketching would be the better way to query. But later I realized drawing with a mouse is actually pretty awkward. I just couldn't sketch the patterns I had in mind, which I felt kind of frustrating.*"

Regarding preciseness, more than half of the participants (10/16) reported that NL offered greater preciseness than mouse-based sketching for specifying query intents in our study setup, consistent with satisfaction scores. As S6 stated, "*NL allows me to directly specify features, which feels much more precise.*" Four (S3, S10–S12) noted that sketch-only approaches restrict intent expression, while others (S9, S15, S16) reported frequent adjustments due to sketch impreciseness. Seven (7/16) also reported NL limitations: "*it is effortful to use*" (S3, S7, S13), "*typing is inconvenient*" (S4, S6), and "*the more complex a pattern is, the more NL needed to describe it*" (S1, S11). Six (S1, S5, S7, S8, S13, S14) further noted NL's cognitive burden, as they "*need to think about how to phrase it*"; two (S5, S11) occasionally struggled to describe patterns clearly.

7.3.3 By-Example NL Authoring. Most participants (11/16) appreciated the by-example function, describing it as "*efficient*" (S3, S4, S15), as helping formulate clear queries without crafting wording (S5, S7, S8, S14, S15), and as supporting multi-relation conditions (S1, S4, S13, S16). Four (S2, S6, S11, S13) noted it helped mainly when a closely matched example was available.

Preferences were pattern- and stage-dependent: More than half of the participants (10/16) preferred manual NL for simple patterns but by-example for complex ones; as S9 explained, "*For simple patterns, I prefer typing natural language. For complex trends or relationships, I'd use by-example mode. It's faster than typing everything out manually.*" Six (S1–S4, S8, S11) preferred manual NL for initial formulation but by-example for later refinement.

7.3.4 Query-Interpretation Transparency and Trust. Most participants (12/16) reported that glyphs helped them understand and validate query processing. S2 mentioned, "*The system (ChronoMantic) provides corresponding text highlights that match the glyphs, letting me verify whether the generated query conditions align with my intent. It's like a built-in validation step.*" S7 stated, "*The glyph design is clean and self-explanatory. The meanings of glyphs are immediately understandable at first glance.*" Three (S2, S6, S15) reported trust in *ChronoMantic*; as S15 noted, "*Text-based descriptions feel more reliable than sketching.*" However, five (S3, S5, S8, S12, S14) raised concerns, mainly about complex NL queries (S3, S5, S12); S12 said, "*I'm concerned that for highly complex queries, ChronoMantic might struggle to comprehend them.*" For *Qetch*, three (S2, S3, S10) reported limited interpretability; as S3 observed, "*Qetch offers poorer explainability. Users must reverse-engineer why certain results matched the sketch.*" Seven expressed distrust toward *Qetch*, citing sketch accuracy (S2, S12, S15) or intent capture (S8, S15, S16).

7.3.5 Suggestions for Improvement. Participants suggested direct glyph editing (S1–S4, S12), adding sketch-based querying (S2, S3, S8, S14), and enabling point (rather than segment) selection for value relations in by-example refinement (S6, S12, S16).

8 Discussion

In this section, we mainly reflect on design lessons for natural language interfaces and the query specification grammar, summarize observed failure cases and potential mitigations, and discuss limitations and future work.

8.1 Lessons Learned

Based on our study findings, we derive broader design implications for future NL-driven interactive interfaces.

Lesson 1: Interpretable visual feedback can enhance user trust in imperfect NL systems. Participants described the visual glyph as a "built-in validation step" and "self-explanatory" (Section 7.3). This aligns with Cai et al. [10] on refinement tools for "imperfect algorithms": they help users "develop a mental model of the ML algorithm" and "increase user trust." In *ChronoMantic*, the LLM parser (Section 4.3) is inherently imperfect and may fail under semantic ambiguity (Section 8.3); the glyph externalizes parsed trend conditions, relations, and attributes, helping users compare the system's interpretation with their intent and correct misinterpretations before execution. Together, these findings suggest that, to support user trust, NL interfaces should expose inspectable, condition-level representations of system interpretations rather than only final outputs.

Lesson 2: Interaction modalities serve complementary, task-dependent roles. Participants valued sketching for its "*convenience*" and "*intuitiveness*", and NL for its "*preciseness*" (Section 7.3). By-example interaction further supported refinement from observed results without requiring users to reformulate the query. Together, these findings suggest that future interfaces could assign complementary roles to different modalities. For example, a multimodal workflow could allow users to begin with a coarse sketch, add semantic and relational constraints through NL, refine the query using observed examples, and use direct editing for precise repair.

8.2 Query Specification Grammar

A query grammar defines the range and complexity of patterns users can express and provides a structured basis for inspecting, verifying, and refining the system's interpretation. *ChronoMantic*'s trend-based grammar translates temporal NL into explicit trend units, attributes, and relations for geometric patterns. Compared with the trend-oriented specifications in *Qetch* [43] and *ShapeSearch* [60], it explicitly supports relational conditions (e.g., one peak being taller than the preceding one), helping bridge high-level intent and low-level specifications. The LLM can implicitly compose multiple conjunctive conditions from NL, allowing users to articulate compound queries without writing explicit logical syntax. However, the current grammar does not support OR/NOT and has limited handling of precise numerical constraints. A detailed comparison of logical operator support among *ChronoMantic*, *Qetch*, and *ShapeSearch* is provided in the supplementary material. Despite these limitations, the grammar is extensible to more complex temporal semantics (e.g., periodicity, stochastic behavior, or multivariate dependencies) through additional fields and corresponding computational feature functions, following feature-based approaches such as *Qute* [29].

8.3 Failure Cases

We observed failures mainly in NL parsing and by-example authoring and discuss them with mitigation strategies.

8.3.1 Failures of NL Parsing. We identified three types:

- **Semantic ambiguity in user queries.** Ambiguous expressions can be misinterpreted (e.g., "plateau" intended as "rise-steady-fall" parsed as a flat trend), reflecting a known challenge in NL data interfaces: bridging the gap between linguistic expression and intent [11]. A potential mitigation is an LLM-agent ambiguity detector that prompts clarification when multiple interpretations are plausible, aligning execution with user intent.
- **Hallucination in domain-specific patterns.** Despite domain knowledge (e.g., *bullish divergence*), LLMs may fail to translate concepts into correct structured patterns, understanding *what* a concept is but not *how* it is composed. We plan parameter-efficient fine-tuning (e.g., LoRA [26]) on "domain term"–"structured pattern" pairs to improve semantic-to-structure mapping.
- **Limited numerical comprehension of LLMs.** LLMs often mishandle precise quantities (e.g., "*a flat trend followed by 20 rising trends*" producing a number other than 20). We will extract numerical constraints from user input rather than relying on LLMs to generate precise quantities.

8.3.2 Failures of By-Example Authoring. We identified two types:

- **Limitations in specifying intents.** By-example authoring depends on choosing appropriate approximation levels; if the intended pattern cannot be specified at the current level, users must retry. We plan to support more flexible example specification, e.g., direct glyph editing to create desired examples.
- **Inability to transform examples to appropriate NL.** *ChronoMantic* may generate incorrect/suboptimal NL (e.g., referring to the second rise as the "third rising trend") and overly verbose descriptions for multi-trend patterns. We will improve example-to-NL generation via prompt engineering: first produce detailed, precise descriptions, then compress them into concise queries.

8.4 Limitations and Future Work

We outline limitations and directions for improving both functionality and evaluation.

8.4.1 Functionality. We mainly discuss the following four aspects:

Retrieval robustness and efficiency. The current piecewise-linear approximation remains sensitive to high levels of noise. Future work will explore alternative approximation and feature-extraction methods, together with adaptive method selection, to improve robustness. We also plan to investigate domain-specific pattern pre-mining (e.g., in finance) to improve retrieval efficiency and support cold-start exploration.

Retrieval transparency and uncertainty. *ChronoMantic* currently provides query-interpretation transparency only: linked glyphs and text highlights expose parsed conditions and their mappings to the query text. It does not explain why individual candidates are included or excluded, or how approximation levels, segmentation, thresholds, and relational constraints affect retrieval. Future work could extend retrieval and parameter transparency through candidate-level explanations, sensitivity views showing how results change under alternative settings, and uncertainty cues that flag borderline or unstable matches.

Retrieval correctness and coverage. The result and detail panels, together with sorting and filtering, support post-hoc inspection of returned matches and help users identify apparent false positives. However, the current system does not surface potential false negatives; users can only probe possible misses by rephrasing or relaxing queries and comparing result sets. Future versions could suggest progressive relaxations of query conditions, surface candidates excluded just beyond thresholds, and rank ambiguous candidates using perceptual-relevance estimates, potentially including relevance scores from vision-language models. These mechanisms could help users assess matches and discover possible omissions.

Result-level analysis. Beyond inspecting individual matches, *ChronoMantic* lacks higher-level analysis and summarization for exploratory use (e.g., summarizing the trends that typically follow retrieved instances of "head-and-shoulders" patterns). Future work will add aggregate summaries to support deeper exploration.

8.4.2 Evaluation. Our evaluation can be strengthened in four respects. (1) The 6-person formative interview and 16-person user study limit generalizability; the formative findings informed design rather than population-level claims, and larger, more diverse samples are needed. (2) Stage 1 used keyboard input for NL and mouse input for sketching, while Stage 2 used short simulation tasks; future studies should examine alternative input modalities and more realistic exploratory settings. (3) *Qetch* is a representative open-source baseline rather than an exhaustive benchmark, and the workflow-level comparison does not isolate the effects of individual components; broader baselines (e.g., NL-based or multimodal), component ablations, and an equivalent direct-manipulation baseline remain future work. (4) The sensitivity and parser-robustness checks were lightweight. Future sensitivity analyses could examine alternative segmentation and approximation methods. Parser robustness could be evaluated on ambiguous and domain-specific inputs across diverse phrasings, temporal semantics, and models.

9 Conclusion

This paper presented *ChronoMantic*, an interactive approach for constructing, interpreting, and refining NL-specified time series pattern queries. Informed by formative interviews with six time series analysts, *ChronoMantic* integrates glyph-based feedback on query interpretation and by-example refinement for iterative querying. We illustrated its workflow through two usage scenarios, evaluated its retrieval performance on a manually annotated benchmark, and assessed its query effectiveness and usability through a two-stage user study. Participants reported high satisfaction with its query results and valued its explanatory glyphs and by-example refinement. Future work will broaden the supported temporal representations, improve retrieval robustness and transparency, and evaluate the system in more realistic exploratory settings.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62532011 and 62402421, and by the Ningbo Yongjiang Talent Programme under Grant 2024A-399-G.

References

[1] Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zaït. 1995. Querying Shapes of Histories. In *Proc. Int. Conf. Very Large Data Bases*. Morgan Kaufmann Publishers Inc., 502–514.

[2] Amazon Web Services. 2024. Amazon Timestream. https://aws.amazon.com/cn/timestream/. Last accessed on 2025-11-24.

[3] Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. 1995. Natural Language Interfaces to Databases - An Introduction. *Nat. Lang. Eng.* 1, 1 (1995), 29–81. doi:10.1017/S135132490000005X

[4] Anthropic. 2024. Claude. https://www.anthropic.com/claude. Last accessed on 2025-11-24.

[5] Tom Bamford, Andrea Coletta, Elizabeth Fons, Sriram Gopalakrishnan, Svitlana Vyetrenko, Tucker Balch, and Manuela Veloso. 2023. Multi-Modal Financial Time-Series Retrieval Through Latent Space Projections. In *Proc. ACM Int. Conf. AI Finance*. 498–506. doi:10.1145/3604237.3626901

[6] Alexander Bendeck, Dennis Bromley, and Vidya Setlur. 2024. SlopeSeeker: A Search Tool for Exploring a Dataset of Quantifiable Trends. In *Proc. ACM Int. Conf. Intell. User Interfaces*. 817–836. doi:10.1145/3640543.3645208

[7] Dennis Bromley and Vidya Setlur. 2023. What Is the Difference Between a Mountain and a Molehill? Quantifying Semantic Labeling of Visual Features in Line Charts. In *Proc. IEEE Vis. Conf. (VIS)*. 161–165. doi:10.1109/VIS54172.2023.00041

[8] John Brooke. 1996. SUS: A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*. CRC Press, 207–212. doi:10.1201/9781498710411-35

[9] Paolo Buono, Aleks Aris, Catherine Plaisant, Amir Khella, and Ben Shneiderman. 2005. Interactive Pattern Search in Time Series. *Visualization and Data Analysis 2005* 5669, 1 (2005), 175–186. doi:10.1117/12.587537

[10] Carrie J. Cai, Emily Reif, Narayan Hegde, Jason Hipp, Been Kim, Daniel Smilkov, Martin Wattenberg, Fernanda Viegas, Greg S. Corrado, Martin C. Stumpe, and Michael Terry. 2019. Human-Centered Tools for Coping with Imperfect Algorithms During Medical Decision-Making. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.* 1–14. doi:10.1145/3290605.3300234

[11] Robin Shing Moon Chan, Rita Sevastjanova, and Mennatallah El-Assady. 2026. PleaSQLarify: Visual Pragmatic Repair for Natural Language Database Querying. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.* 1–18. doi:10.1145/3772318.3791265

[12] Zhutian Chen, Qisen Yang, Xiao Xie, Johanna Beyer, Haijun Xia, Yingcai Wu, and Hanspeter Pfister. 2023. Sporthesia: Augmenting Sports Videos Using Natural Language. *IEEE Trans. Vis. Comput. Graph.* 29, 1 (2023), 918–928. doi:10.1109/TVCG.2022.3209497

[13] Michael Correll and Michael Gleicher. 2016. The Semantics of Sketch: Flexibility in Visual Query Systems for Time Series Data. In *Proc. IEEE Conf. Vis. Anal. Sci. Technol.* 131–140. doi:10.1109/VAST.2016.7883519

[14] Per-Erik Danielsson. 1980. Euclidean Distance Mapping. *Comput. Graph. Image Process.* 14, 3 (1980), 227–248. doi:10.1016/0146-664X(80)90054-4

[15] DeepSeek-AI. 2025. DeepSeek-R1. https://huggingface.co/deepseek-ai/DeepSeek-R1. Accessed on 2025-04-10.

[16] Chaoran Fan, Krešimir Matković, and Helwig Hauser. 2021. Sketch-Based Fast and Accurate Querying of Time Series Using Parameter-Sharing LSTM Networks. *IEEE Trans. Vis. Comput. Graph.* 27, 12 (2021), 4495–4506. doi:10.1109/TVCG.2020.3002950

[17] Tinghao Feng, Yueqi Hu, Jing Yang, Tom Polk, Ye Zhao, Shixia Liu, and Zhaocong Yang. 2023. TimePool: Visually Answer "Which and When" Questions On Univariate Time Series. In *Proc. IEEE Vis. Conf. (VIS)*. 201–205. doi:10.1109/VIS54172.2023.00049

[18] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proc. ACM Symp. User Interface Softw. Technol.* 489–500. doi:10.1145/2807442.2807478

[19] Sneha Gathani, Peter Lim, and Leilani Battle. 2020. Debugging Database Queries: A Survey of Tools, Techniques, and Users. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.* 1–16. doi:10.1145/3313831.3376485

[20] Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Hanspeter Pfister, and Marc Streit. 2013. LineUp: Visual Analysis of Multi-Attribute Rankings. *IEEE Trans. Vis. Comput. Graph.* 19, 12 (2013), 2277–2286. doi:10.1109/TVCG.2013.173

[21] Hua Guo, Steven R. Gomez, Caroline Ziemkiewicz, and David H. Laidlaw. 2016. A Case Study Using Visualization Interaction Logs and Insight Metrics to Understand How Analysts Arrive at Insights. *IEEE Trans. Vis. Comput. Graph.* 22, 1 (2016), 51–60. doi:10.1109/TVCG.2015.2467613

[22] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*. Vol. 52. North-Holland, 139–183. doi:10.1016/S0166-4115(08)62386-9

[23] Harry Hochheiser. 2002. Interactive Querying of Time Series Data. In *Ext. Abstr. CHI Conf. Hum. Factors Comput. Syst.* 552–553. doi:10.1145/506443.506477

[24] Christian Holz and Steven Feiner. 2009. Relaxed Selection Techniques for Querying Time-Series Graphs. In *Proc. ACM Symp. User Interface Softw. Technol.* 213–222. doi:10.1145/1622176.1622217

[25] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2018. Applying Pragmatics Principles for Interaction with Visual Analytics. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 309–318. doi:10.1109/TVCG.2017.2744684

[26] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proc. Int. Conf. Learn. Represent.* https://openreview.net/forum?id=nZeVKeeFYf9

[27] Jieying Huang, Yang Xi, Junnan Hu, and Jun Tao. 2023. FlowNL: Asking the Flow Data in Natural Languages. *IEEE Trans. Vis. Comput. Graph.* 29, 1 (2023), 1200–1210. doi:10.1109/TVCG.2022.3209453

[28] Yanwei Huang, Yunfan Zhou, Ran Chen, Changhao Pan, Xinhuan Shu, Di Weng, and Yingcai Wu. 2024. Interactive Table Synthesis With Natural Language. *IEEE Trans. Vis. Comput. Graph.* 30, 9 (2024), 6130–6145. doi:10.1109/TVCG.2023.3329120

[29] Shima Imani, Sara Alaee, and Eamonn Keogh. 2021. Qute: Query by Text Search for Time Series Data. In *Proc. Future Technol. Conf.* Springer, 412–427. doi:10.1007/978-3-030-63089-8_27

[30] InfluxData. 2024. InfluxDB Time Series Data Platform. https://www.influxdata.com/. Last accessed on 2025-11-24.

[31] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I. Webb, Irwin King, and Shirui Pan. 2024. A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 46, 12 (2024), 10466–10485. doi:10.1109/TPAMI.2024.3443141

[32] Eamonn J. Keogh and Padhraic Smyth. 1997. A Probabilistic Approach to Fast Pattern Matching in Time Series Databases. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* AAAI Press, 24–30.

[33] Dae Hyun Kim, Enamul Hoque, and Maneesh Agrawala. 2020. Answering Questions about Charts and Generating Visual Explanations. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.* 1–13. doi:10.1145/3313831.3376467

[34] Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural Language to SQL: Where Are We Today? *Proc. VLDB Endowment* 13, 10 (2020), 1737–1750. doi:10.14778/3401960.3401970

[35] KX. 2024. kdb+. https://kx.com/products/kdb/. Last accessed on 2025-11-24.

[36] Fritz Lekschas, Brant Peterson, Daniel Haehn, Eric Ma, Nils Gehlenborg, and Hanspeter Pfister. 2020. Peax: Interactive Visual Pattern Search in Sequential Data Using Unsupervised Deep Representation Learning. *Comput. Graph. Forum* 39, 3 (2020), 167–179. doi:10.1111/cgf.13971

[37] James R Lewis and Jeff Sauro. 2009. The Factor Structure of the System Usability Scale. In *Proc. Int. Conf. Hum. Centered Des.* Springer, 94–103. doi:10.1007/978-3-642-02806-9_12

[38] Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan-chi Chiu. 2003. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*. 2–11. doi:10.1145/882082.882086

[39] Shuhan Liu, Yuan Tian, Zikun Deng, Weiwei Cui, Haidong Zhang, Di Weng, and Yingcai Wu. 2025. Relation-driven Query of Multiple Time Series. *IEEE Trans. Vis. Comput. Graph.* 31, 8 (2025), 4210–4225. doi:10.1109/TVCG.2024.3397554

[40] Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021. Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*. 1235–1247. doi:10.1145/3448016.3457261

[41] Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2022. Natural Language to Visualization by Neural Machine Translation. *IEEE Trans. Vis. Comput. Graph.* 28, 1 (2022), 217–226. doi:10.1109/TVCG.2021.3114848

[42] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proc. Annu. Meet. Assoc. Comput. Linguist.* 55–60. doi:10.3115/V1/P14-5010

[43] Miro Mannino and Azza Abouzied. 2018. Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.* 1–13. doi:10.1145/3173574.3173962

[44] MeasuringU. 2011. SUS - System Usability Scale. https://measuringu.com/sus/. Last accessed on 2025-11-24.

[45] Rishab Mitra, Arpit Narechania, Alex Endert, and John T. Stasko. 2022. Facilitating Conversational Interaction in Natural Language Interfaces for Visualization. In *Proc. IEEE Vis. Conf. (VIS)*. 6–10. doi:10.1109/VIS54862.2022.00010

[46] Meinard Müller. 2007. Dynamic Time Warping. In *Information Retrieval for Music and Motion*. Springer, 69–84. doi:10.1007/978-3-540-74048-3_4

[47] Prithiviraj K. Muthumanickam, Katerina Vrotsou, Matthew Cooper, and Jimmy Johansson. 2016. Shape Grammar Extraction for Efficient Query-by-Sketch Pattern Matching in Long Time Series. In *Proc. IEEE Conf. Vis. Anal. Sci. Technol.* 121–130. doi:10.1109/VAST.2016.7883518

[48] Arpit Narechania, Adam Fourney, Bongshin Lee, and Gonzalo A. Ramos. 2021. DIY: Assessing the Correctness of Natural Language to SQL Systems. In *Proc. ACM Int. Conf. Intell. User Interfaces*. 597–607. doi:10.1145/3397481.3450667

[49] Arpit Narechania, Arjun Srinivasan, and John T. Stasko. 2021. NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries. *IEEE Trans. Vis. Comput. Graph.* 27, 2 (2021), 369–379. doi:10.1109/TVCG.2020.3030378

[50] OpenAI. 2024. ChatGPT. https://openai.com/chatgpt/overview/. Last accessed on 2025-11-24.

[51] Pallets Team. 2025. Flask Documentation. https://flask.palletsprojects.com/en/stable/. Last accessed on 2025-11-24.

[52] Prompting Guide. 2025. Few-Shot Prompting Techniques. https://www.promptingguide.ai/techniques/fewshot. Last accessed on 2025-11-24.

[53] React Team. 2025. React. https://react.dev/. Last accessed on 2025-11-24.

[54] Nicholas Ruta, Naoko Sawada, Katy McKeough, Michael Behrisch, and Johanna Beyer. 2019. SAX Navigator: Time Series Exploration through Hierarchical Clustering. In *Proc. IEEE Vis. Conf. (VIS)*. 236–240. doi:10.1109/VISUAL.2019.8933618

[55] Kathy Ryall, Neal Lesh, Tom Lanning, Darren Leigh, Hiroaki Miyashita, and Shigeru Makino. 2005. QueryLines: Approximate Query for Visual Browsing. In *Ext. Abstr. CHI Conf. Hum. Factors Comput. Syst.* 1765–1768. doi:10.1145/1056808.1057017

[56] Stan Salvador and Philip Chan. 2007. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intell. Data Anal.* 11, 5 (2007), 561–580. doi:10.3233/IDA-2007-11508

[57] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017), 341–350. doi:10.1109/TVCG.2016.2599030

[58] Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A Natural Language Interface for Visual Analysis. In *Proc. ACM Symp. User Interface Softw. Technol.* 365–377. doi:10.1145/2984511.2984588

[59] Vidya Setlur, Enamul Hoque, Dae Hyun Kim, and Angel X. Chang. 2020. Sneak Pique: Exploring Autocompletion as a Data Discovery Scaffold for Supporting Visual Analysis. In *Proc. ACM Symp. User Interface Softw. Technol.* 966–978. doi:10.1145/3379337.3415813

[60] Tarique Siddiqui, Paul Luh, Zesheng Wang, Karrie Karahalios, and Aditya Parameswaran. 2020. ShapeSearch: A Flexible and Efficient System for Shape-Based Exploration of Trendlines. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*. 51–65. doi:10.1145/3318464.3389722

[61] Arjun Srinivasan and John T. Stasko. 2018. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 511–521. doi:10.1109/TVCG.2017.2745219

[62] Tableau. 2023. Time Series Analysis: Definition, Types, Techniques, and When It's Used. https://www.tableau.com/learn/articles/time-series-analysis. Last accessed on 2025-11-24.

[63] Timescale. 2024. TimescaleDB: An Open-Source Time-Series SQL Database Optimized for Fast Ingest and Complex Queries. https://github.com/timescale/timescaledb. Last accessed on 2025-11-24.

[64] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.* 1–17. doi:10.1145/3613904.3642639

[65] Andrew Villca-Rocha, Max Zheng, Chengzhu Duan, and Hongning Wang. 2021. Towards Semantic Search in Building Sensor Data. In *Proc. ACM Int. Conf. Syst. Energy-Effic. Built Environ.* 164–167. doi:10.1145/3486611.3486647

[66] Chenglong Wang, Bongshin Lee, Steven M. Drucker, Dan Marshall, and Jianfeng Gao. 2025. Data Formulator 2: Iterative Creation of Data Visualizations, with AI Transforming Data Along the Way. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.* 1–17. doi:10.1145/3706598.3713296

[67] Chenglong Wang, John Thompson, and Bongshin Lee. 2024. Data Formulator: AI-Powered Concept-Driven Visualization Authoring. *IEEE Trans. Vis. Comput. Graph.* 30, 1 (2024), 1128–1138. doi:10.1109/TVCG.2023.3326585

[68] Yun Wang, Zhitao Hou, Leixian Shen, Tongshuang Wu, Jiaqi Wang, He Huang, Haidong Zhang, and Dongmei Zhang. 2023. Towards Natural Language-Based Visualization Authoring. *IEEE Trans. Vis. Comput. Graph.* 29, 1 (2023), 1222–1232. doi:10.1109/TVCG.2022.3209357

[69] Martin Wattenberg. 2001. Sketching a Graph to Query a Time-Series Database. In *Ext. Abstr. CHI Conf. Hum. Factors Comput. Syst.* 381–382. doi:10.1145/634067.634292

[70] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] doi:10.48550/arXiv.2505.09388

[71] Bowen Yu and Cláudio T. Silva. 2020. FlowSense: A Natural Language Interface for Visual Data Exploration within a Dataflow System. *IEEE Trans. Vis. Comput. Graph.* 26, 1 (2020), 1–11. doi:10.1109/TVCG.2019.2934668

[72] Yuncong Yu, Dylan Kruyff, Jiao Jiao, Tim Becker, and Michael Behrisch. 2023. PSEUDo: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback. *IEEE Trans. Vis. Comput. Graph.* 29, 1 (2023), 33–42. doi:10.1109/TVCG.2022.3209431

[73] Yue Zhao, Yunhai Wang, Jian Zhang, Chi-Wing Fu, Mingliang Xu, and Dominik Moritz. 2022. KD-Box: Line-Segment-Based KD-Tree for Interactive Exploration of Large-Scale Time-Series Data. *IEEE Trans. Vis. Comput. Graph.* 28, 1 (2022), 890–900. doi:10.1109/TVCG.2021.3114865