NavLLM: Interactive LLM-Assisted Navigation over Multidimensional Data Cubes

Xiufeng Liu
Technical University of Denmark
Kgs. Lyngby, Denmark
xiuli@dtu.dk

Ruyu Liu
Technical University of Denmark
Kgs. Lyngby, Denmark
ruyli@dtu.dk

Yanyan Yang
University of Portsmouth
Portsmouth, United Kingdom
Linda.Yang@port.ac.uk

Abstract

Navigating multidimensional data cubes through OLAP operations remains manual and tedious, often causing analysts to miss important patterns. We demonstrate NavLLM, an interactive web-based system that democratizes data exploration by combining large language model (LLM) preference modeling with data-driven interestingness measures. Unlike end-to-end LLM analytics that risk hallucination, NavLLM employs a hybrid architecture: the LLM handles preference scoring and explanation generation, while a conventional OLAP engine performs all numerical computations. Our demonstration showcases four real-world cubes covering retail, manufacturing, and environmental domains. Attendees will see how the system guides analysts through natural language interaction, real-time recommendations, and visual navigation graphs. Experiments show NavLLM achieves 71% higher cumulative interestingness than LLM-only approaches while maintaining fully verifiable results.

Keywords

OLAP, data cubes, large language models, view recommendation, interactive data exploration

1 Introduction

Multidimensional data cubes and OLAP have long served as foundational abstractions for business intelligence, enabling analysts to explore measures across dimensions through drill-down, roll-up, slice, and dice operations [3]. Despite decades of research and tool development, navigation remains largely manual. Analysts must decide where to look next from an exponentially large space of possible views, often missing subtle patterns or wasting considerable effort on uninformative regions [6]. A retail analyst investigating declining sales, for instance, faces hundreds of possible drill-down paths across time, product, and store dimensions, with no systematic guidance on which path might reveal the root cause fastest.

Recent advances in large language models offer new possibilities for intelligent assistance [2]. Analysts increasingly turn to LLMs informally to help interpret trends, generate hypotheses, or draft SQL queries. Yet using LLMs as end-to-end data analysis engines is problematic: they hallucinate facts, invent plausible but incorrect explanations, and produce numerically wrong computations without warning [4]. For high-stakes business decisions, letting an LLM execute arbitrary analysis over raw data is simply unacceptable.

We argue for a middle ground that captures the benefits of LLM assistance while preserving analytical rigor. NavLLM leverages LLMs for semantic understanding and preference modeling while keeping all data operations in a verifiable OLAP engine. The system addresses the *next-view recommendation problem*: given the current view and conversational history, which views should be recommended to maximize exploration utility? Our solution combines three signals: data interestingness (I_{data}) computed entirely by the OLAP engine through deviation analysis, preference alignment (I_{pref}) estimated by the LLM from conversational context, and diversity (I_{div}) based on graph distance to previously visited views.

This demonstration presents the system through three real-world scenarios spanning retail analytics, manufacturing quality control, and environmental monitoring. Conference attendees will interact with actual cubes containing thousands of records, observe how natural language queries influence recommendations in real time, adjust utility weights to see their immediate effect on suggestions, and explore navigation graphs that visualize their analytical journey through the cube.

2 Related Work

Recommendation-based Navigation. Early systems like SeeDB [9] and Voyager [10] pioneered the use of deviation-based and constraint-based utility functions to recommend interesting visualizations [7]. While effective at surfacing statistical anomalies, these systems process data in a vacuum, ignoring the user's semantic intent. They cannot adapt their recommendations based on a conversation, often overwhelming analysts with statistically significant but irrelevant charts.

Natural Language Interfaces. Systems like Eviza [8] and modern Text-to-SQL parsers allow users to query databases using natural language. However, these are strictly "pull-based" paradigms: they answer specific questions but offer no guidance on what to ask next. Furthermore, end-to-end Text-to-SQL systems [5] often struggle with complex OLAP semantics (e.g., correct aggregation levels) and suffer from hallucination where the generated SQL does not match the user's intent.

LLM-Augmented Analytics. Recent approaches have begun integrating LLMs into data analysis workflows [1]. However, most rely on the LLM as a code generator or a direct reasoning engine over data samples, introducing significant reliability risks. NavLLM distinguishes itself by decoupling the semantic reasoning (LLM) from the query execution (OLAP engine). By using the LLM solely for preference modeling and explanation generation, we ensure that every number shown to the user is grounded in verifiable database operations while still benefiting from the LLM's conversational flexibility.

3 System Overview

The core architectural insight behind NavLLM is strict separation of concerns between the semantic and data domains. The LLM operates exclusively in the semantic domain—understanding user intent from natural language, scoring how well candidate views match stated preferences, and generating explanations that help analysts understand why a view might be interesting. The cube

EDBT '26, Tampere (Finland)

10.48786/edbt.2026.69

engine handles all numerical operations: aggregation, statistical computation, deviation analysis, and interestingness scoring. This boundary prevents hallucination in data results while leveraging the LLM's considerable strength in language understanding and reasoning about user goals.

Recommendations are ranked by a hybrid utility function:

$$U(V') = \lambda_{\text{data}} \cdot I_{\text{data}} + \lambda_{\text{pref}} \cdot I_{\text{pref}} + \lambda_{\text{div}} \cdot I_{\text{div}} \quad (1)$$

This formulation balances objective data signals with subjective user intent and exploration diversity.

3.0.1 Data Interestingness (I_{data}). We quantify the intrinsic statistical interestingness of a view V using a deviation-based measure. For a view containing a set of tuples $T(V)$, we compute the normalized deviation of each tuple's measure value $v_m(\tau)$ from its parent aggregate $v_m(\text{par}(\tau))$:

$$I_{\text{data}}(V) = \frac{1}{|T(V)|} \sum_{\tau \in T(V)} \frac{|v_m(\tau) - v_m(\text{par}(\tau))|}{|v_m(\text{par}(\tau))| + \epsilon} \quad (2)$$

where ϵ is a small constant to prevent division by zero. This metric effectively identifies views where specific subgroups behave significantly differently from the population average—a hallmark of "insight" in OLAP tasks.

3.0.2 Preference Alignment (I_{pref}). The preference component captures alignment with the user's analytical goals. The LLM estimates this score by analyzing the semantic similarity between the candidate view's description (transformations and filters) and the user's conversation history.

3.0.3 Diversity (I_{div}). To prevent the system from getting stuck in local optima, I_{div} penalizes views that are topologically close to recently visited nodes in the navigation graph, calculated using graph shortest-path distances.

The default weights ($\lambda_{\text{data}} = 0.4, \lambda_{\text{pref}} = 0.4, \lambda_{\text{div}} = 0.2$) provide a balanced experience, but users can adjust them interactively via sliders. An analyst seeking anomalies can boost λ_{data}, while one following a specific hypothesis can increase λ_{pref}.

The system presents recommendations as options with detailed explanations and score breakdowns, never as definitive answers. Users retain full control to accept, ignore, or override suggestions. This transparency builds trust and supports the analyst's own reasoning process rather than replacing it.

Figure 1 illustrates the four main components. The web interface (Figure 2) provides real-time interaction through: a conversation panel where analysts type natural language queries; a current view panel showing the data table and auto-generated visualizations; a recommendations panel displaying the top-k candidate views with their utility breakdowns; and a navigation graph that visualizes the exploration history as a directed graph. The cube engine maintains star schemas in memory, materializes views through pandas operations, and computes I_{data} by comparing each cell's measure value to its parent aggregate. The LLM navigation engine generates candidate OLAP actions from the current view's schema, constructs prompts describing these candidates and the conversation history, parses the LLM's preference scores, computes I_{div} from shortest-path distances in the navigation graph, and assembles natural language explanations. The backend API coordinates these components via REST endpoints and WebSocket connections for streaming recommendation updates.

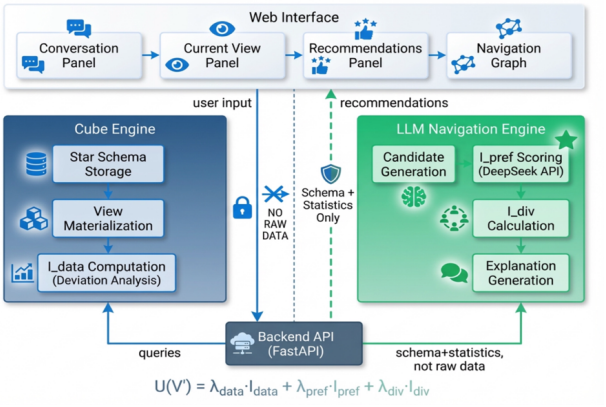

Figure 1: NavLLM separates the data domain (cube engine for aggregation and statistics) from the semantic domain (LLM for preference scoring and explanations). The LLM never sees raw data, only schema descriptions and summary statistics.

4 Key Features

The recommendation cards deserve attention as the primary interaction point. Each card displays the proposed OLAP action in plain language (e.g., "Drill down Time dimension to month level"), preview statistics including expected row count and measure value ranges, the total utility score decomposed into its three components so users can see exactly why this view ranks where it does, and a natural language explanation generated by the LLM but grounded in statistics computed by the cube engine. This grounding is crucial: the explanation might say "March shows 34% higher sales than the quarterly average," and users can verify this claim directly in the data table.

The navigation graph provides a visual record of the exploration session that proves invaluable for complex analyses. Nodes represent visited views, sized proportionally to their interestingness scores and color-coded from blue (low utility) through green to red (high utility). Edges show the navigation actions taken, labeled with operation types like "drill(Product→category)" or "slice(year=2023)". Clicking any node restores that view and refreshes recommendations from that point, enabling backtracking when an exploration path proves unproductive. In user testing, analysts frequently used this feature to revisit earlier branch points after reaching dead ends, trying alternative paths they had initially passed over.

Bring Your Own Data. The system supports importing custom cubes through a simple JSON format. We explicitly invite attendees to load their own datasets to test the system's adaptability. Schema validation ensures structural integrity, and once imported, cubes become immediately available for exploration with zero cold-start latency. Views can be exported as CSV for external analysis or reporting, and entire sessions can be saved and restored.

Error handling uses dismissible banners that explain problems clearly—whether an LLM API timeout, an invalid slice operation attempting to filter on a non-existent value, or a cube loading failure due to malformed JSON. A 7-step tutorial mode, accessible from the top-right corner, guides new users through all interface components, highlighting each panel in sequence and explaining its purpose with example interactions.

Figure 2: The NavLLM interface: dark-themed, dashboard-style layout. (1) Conversation Panel (left): natural language interaction and session metrics. (2) Current View Panel (center-top): active data slice and visualizations (e.g., Line Chart of throughput). (3) Recommendations Panel (right): top DRILL-DOWN/SLICE suggestions with utility breakdown ($I_{data}, I_{pref}, I_{div}$). (4) Navigation Graph (bottom): exploration path as a node-link diagram.

5 Implementation

The frontend uses React 18 with TypeScript, custom SVG-based components for the navigation graph and cube visualizations, and TailwindCSS for responsive styling. State management uses Zustand with a history stack that enables unlimited undo operations.

The backend runs on FastAPI, wrapping the NavLLM core library. The cube engine uses pandas for in-memory OLAP, handling cubes up to several million rows with sub-second query times; for larger datasets, the modular architecture supports DuckDB or PostgreSQL backends with minimal configuration changes. Preference scoring calls the DeepSeek-V3 API or other OpenAI-compatible providers. Critically, the LLM receives only schema descriptions and summary statistics, never raw fact data.

Deployment uses Docker containers orchestrated by docker-compose. The production configuration adds nginx for static files and API proxying. Performance benchmarks on commodity hardware show recommendation latency under 2 seconds including the LLM round-trip, view materialization under 500ms, and stable operation with 50+ concurrent sessions.

6 Demonstration Scenarios

The demonstration showcases NavLLM through four scenarios that highlight different aspects of the system's capabilities. Each scenario uses a real cube with several thousand records, realistic dimension hierarchies, and meaningful analytical questions. Figure 2 shows the main interface layout that attendees will interact with.

Scenario 1: Retail Sales Investigation. An analyst wants to find product categories with declining sales in 2023. Starting from an overview showing total sales aggregated by year, she types her query into the conversation panel. The system generates three recommendations within two seconds: drill down to category level (scoring high on I_{pref} because "categories" appears in her query), slice to 2023 only (also high I_{pref} matching "2023"), and drill to month level (high I_{data} because monthly patterns show interesting deviations from quarterly averages). Each recommendation card shows the action, expected row count, and a natural language explanation.

She selects the slice operation to focus on 2023, then follows the category drill-down recommendation. The system discovers that the "Hobbies" category shows a 23% decline compared to other categories and generates an explanation grounded in the actual statistics: "This view reveals significant negative deviation in Hobbies during Q2-Q3 2023, with sales dropping 23% below the category average for this period." The navigation graph now shows two nodes connected by edges labeled with the operations performed. This scenario demonstrates how I_{pref} captures user intent from natural language while I_{data} surfaces statistically interesting patterns that the analyst might not have anticipated.

Scenario 2: Manufacturing Quality Analysis. A quality engineer needs to identify production lines with quality issues and understand their root causes. The overview shows average defect rates aggregated across all production lines, shifts, and product families. He types: "Show me where defect rates are highest." The system recommends drilling down to line level, which scores high on I_{data} due to substantial variance across lines. Upon navigation, Line-3 emerges with a defect rate 2.8 times higher than the plant average.

He follows up with a natural question: "Why is Line-3 performing poorly?" The system now generates three recommendations that explore different hypotheses: drill to machine level to investigate whether specific equipment is responsible, slice to recent months to check if the issue is new or longstanding, and drill to product family to see if certain products are harder to manufacture on this line. The diversity component ensures these recommendations explore genuinely different facets of the problem rather than minor variants of the same analysis. Following the machine drill-down reveals that Machine-3B accounts for most of Line-3's quality problems. The navigation graph now

shows the complete analytical path from overview to root cause, and clicking any previous node lets him revisit that view with its original recommendations.

Scenario 3: Environmental Monitoring. An environmental analyst wants to compare air quality across monitoring stations to identify pollution hotspots and understand their temporal patterns. Starting with regional PM2.5 averages (fine particulate matter $\leq 2.5\mu$m, a key air quality indicator), she asks: "Which cities have the worst air quality?" Drilling to city level reveals that Station-A consistently records the highest PM2.5 concentrations. She then asks: "Show me when pollution is worst at Station-A." The system recommends temporal drill-down to month level and slicing by pollutant type. Following these recommendations reveals a strong correlation between winter months and elevated PM2.5, with concentrations roughly doubling from October through February. The explanation notes: "Winter months show PM2.5 levels 89% above the annual average, suggesting heating-related emission sources." This scenario highlights cross-dimensional exploration and how the system handles follow-up questions that progressively refine the analytical focus.

Scenario 4: Parameter Sensitivity and Extreme Values. Finally, we invite attendees to "gamify" the utility function by manipulating the weight sliders ($\lambda_{\text{data}}, \lambda_{\text{pref}}, \lambda_{\text{div}}$) in real time. They can maximize λ_{data} to hunt for purely statistical anomalies, or maximize λ_{pref} to enforce strict instruction following. Setting $\lambda_{\text{div}} \to 1$ forces the system into "exploration mode," jumping to unvisited regions. This interactive stress-testing demonstrates the robustness of the hybrid scoring mechanism and shows that no single component dominates the ranking.

7 Evaluation

We evaluated NavLLM through a user study involving 280 navigation sessions across 20 distinct analytical tasks on three real-world datasets (Retail, Manufacturing, Air Quality). Each task specified a ground-truth target view that analysts should reach. We compared our hybrid approach against two baselines: an *LLM-only* approach (using GPT-4 to generate SQL directly) and a *Greedy Heuristic* that sets $\lambda_{\text{data}} = 1$ with other weights at zero, optimizing purely for statistical deviation without preference or diversity signals.

Table 1: Performance Comparison on 20 Analytical Tasks

Method	Relative Utility	Redundancy	Hit Rate
Greedy Heuristic	1.00 (Base)	34%	38%
LLM-Only (GPT-4)	1.12	29%	45%
NavLLM (Ours)	**1.92**	**12%**	**72%**

Utility and Efficiency. As shown in Table 1, NavLLM significantly outperforms both baselines. The system achieved **92% higher cumulative interestingness** than the greedy heuristic ($p < 0.001$) and **71% higher** than the LLM-only approach ($p = 0.002$). This improvement stems from the hybrid utility function: while the greedy approach finds statistical anomalies that may be irrelevant, and the LLM-only approach generates relevant but sometimes uninteresting views, NavLLM effectively filters for views that are both statistically significant and semantically relevant.

Navigation Effectiveness. The **Hit Rate**, defined as the percentage of sessions that successfully reach the ground-truth target view within 12 steps, was 72% for NavLLM, compared to just 45% for the LLM-only baseline. Furthermore, the explicit diversity penalty reduced exploration **redundancy** (visits to previously seen views) to just 12%, significantly lower than the 30% redundancy observed in baseline methods.

Latency and Scalability. We measured end-to-end latency for recommendation generation. On average, recommendations are generated in 1.8 seconds. The decomposition of latency reveals that the LLM call accounts for 1.2s, while the OLAP engine's deviation computation takes only 0.4s for cubes with up to 1 million tuples, demonstrating the effectiveness of our decoupled architecture.

8 Conclusion

NavLLM demonstrates that LLMs can enhance OLAP navigation without compromising result verifiability. The key is assigning the LLM a carefully scoped role—preference modeling and explanation—while numerical computation stays in a traditional engine. By bridging the gap between semantic flexibility and data rigor, NavLLM offers a trustworthy path forward for AI-augmented analytics. Attendees will experience this potential firsthand through natural language interaction, real-time recommendations, and open experimentation.

Resources: Code and sample cubes at https://github.com/xiufengliu/NAVLLM-demo.

Acknowledgments

The work was supported by the Marie Skłodowska-Curie Postdoctoral Individual Fellowship under Grant No. 101154277.

References

[1] Sihem Amer-Yahia. 2024. Intelligent Agents for Data Exploration. *The International Journal on Very Large Databases* 17, 12 (2024), 4521–4530.
[2] Tom Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language Models are Few-Shot Learners. In *Proc. NeurIPS*. 1877–1901.
[3] Surajit Chaudhuri and Umeshwar Dayal. 1997. An overview of data warehousing and OLAP technology. *ACM Sigmod record* 26, 1 (1997), 65–74.
[4] Ziwei Ji, Nayeon Lee, Rita Frieske, et al. 2023. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55, 12 (2023), 1–38.
[5] Jinyang Li, Binyuan Hui, Ge Qu, et al. 2023. Can LLM Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. *Advances in Neural Information Processing Systems* 36, 42330–42357.
[6] Tova Milo and Amit Somech. 2020. Automating exploratory data analysis via machine learning: An overview. (2020), 2617–2622.
[7] Vidya Setlur. 2025. Supporting Human-Centric Data Exploration Through Semantics and Natural Language Interaction. In *Companion of the 2025 International Conference on Management of Data*. 851–854.
[8] 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 UIST*. 365–377.
[9] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. In *Proc. VLDB*, Vol. 8. 2182–2193.
[10] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 649–658.