LASEK: LLM-Assisted Style Exploration Kit for Geospatial Data[*]

Tarlan Bahadori
University of California, Riverside
Riverside, California
tbaha001@ucr.edu

Sai Sreekar Sarvepalli
University of California, Riverside
Riverside, California
ssarv003@ucr.edu

Ahmed Eldawy
University of California, Riverside
Riverside, California
eldawy@ucr.edu

ABSTRACT

Geospatial data visualization on a map is an essential tool for modern data exploration tools. However, these tools require users to manually configure the visualization style including color scheme and attribute selection, a process that is both complex and domain-specific. Large Language Models (LLMs) provide an opportunity to intelligently assist in styling based on the underlying data distribution and characteristics. This paper demonstrates LASEK, an LLM-assisted visualization framework that automates attribute selection and styling in large-scale spatio-temporal datasets. The system leverages LLMs to determine which attributes should be highlighted for visual distinction and even suggests how to integrate them in styling options improving interpretability and efficiency. We demonstrate our approach through interactive visualization scenarios, showing how LLM-driven attribute selection enhances clarity, reduces manual effort, and provides data-driven justifications for styling decisions.

PVLDB Reference Format:
Tarlan Bahadori, Sai Sreekar Sarvepalli, and Ahmed Eldawy. LASEK: LLM-Assisted Style Exploration Kit for Geospatial Data. PVLDB, 18(12): 5435 - 5438, 2025.
doi:10.14778/3750601.3750690

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/tarlaun/LASEK.

1 INTRODUCTION

We live in an era of data science and data-driven research. The ubiquity of collected data results in wide availability of repositories with hundreds of thousands of datasets. Studies show that about 60% of these datasets have a geospatial component. There has been some recent work to support scalable geospatial visualization [2, 4, 9]. However, these systems require the users to define the visualization style, e.g., attribute colors, which is a tedious task for datasets with tens or hundreds of attributes that the users are not familiar with. This demonstration addresses a common challenge of how to define appropriate styling for visualization that effectively communicates insights within the data.

This demo presents LASEK (LLM-Assisted Style Exploration Kit for Geospatial Data), which extends Spark and Beast [2], to provide scalable visualization for terabytes of geospatial data with various input/output file formats. The approach of LASEK can be helpful with both server-side rendering systems, e.g., UCR-Star [4] and SedonaViz [9], and client-side rendering systems, e.g., OpenLayers and Google Maps APIs. The proposed toolkit helps users navigate large geospatial datasets that contain many attributes with vague names to help users users identify which ones are necessary for their analysis. The database community has explored visualization recommendation [7, 8] but existing work focused on traditional line plots and pie charts and not map visualization. Building on this insight, LASEK addresses the complexities of spatio-temporal data exploration and styling by integrating Large Language Models (LLMs) to intelligently rank and filter attributes, ensuring users focus only on relevant data. This reduces the time spent exploring unnecessary attributes while improving computational efficiency. By reducing dataset size, LASEK optimizes storage and rendering time without sacrificing visualization quality. Additionally, LLM-based styling ensures uniform color schemes and standardized visual encodings, making it easier to interpret different datasets across multiple use cases.

The data flow and main processing steps of LASEK can be seen in figure 1. To optimize LLM usage costs, we've implemented a two-step LLM call architecture that minimizes token count by transferring only essential information at each stage. Our goals for each step can be summarized as follows:

- **Step 1: Attribute Type Inference** Since datasets are often ingested from unstructured text formats, we first process a data sample through the LLM to detect and parse complex attribute formats such as datetime fields and geometry data, generating an enhanced schema that enables proper data interpretation.
- **Step 2: Calculate Statistical Summaries** As large-scale datasets cannot be processed entirely by LLMs due to memory constraints, we generate comprehensive statistical summaries including top-k most frequent values, min/max values, and variance metrics to provide meaningful context for visualization decisions.
- **Step 3: Attribute Recommendation** This step integrates user intent through either interactive natural language prompts or the selection from the automated visualization-worthy attributes based on statistical significance.
- **Step 4: Stlying Generation** The final step obtains styling code from the LLM based on the selected attributes and statistical patterns, then applies it to the entire dataset for visualization.

[*]This work is supported in part by the National Science Foundation (NSF) under grant IIS-2046236

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750690

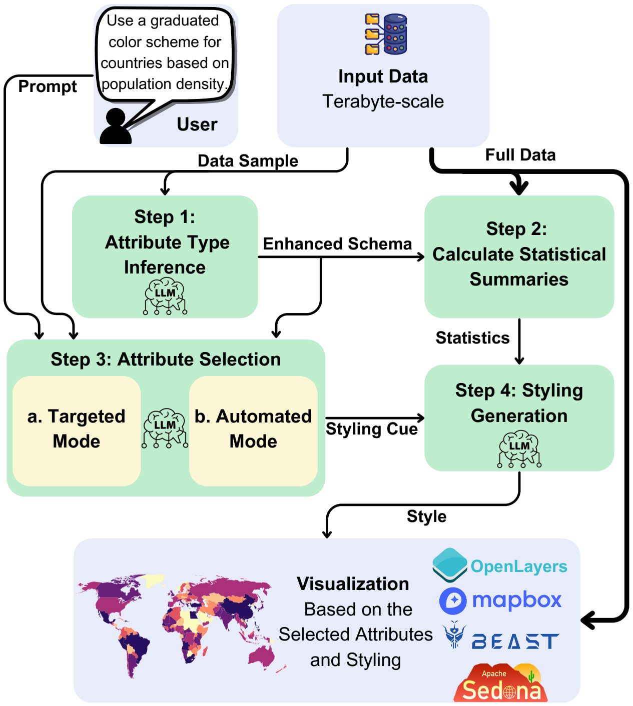

Use a graduated color scheme for countries based on population density.
Prompt
User
Input Data
Terabyte-scale
Data Sample
Full Data
Step 1: Attribute Type Inference
LLM
Enhanced Schema
Step 2: Calculate Statistical Summaries
Statistics
Step 3: Attribute Selection
a. Targeted Mode
LLM
b. Automated Mode
Styling Cue
Step 4: Styling Generation
LLM
Style
Visualization Based on the Selected Attributes and Styling
OpenLayers
mapbox
BEAST
Apache Sedona

Figure 1: Overview of LASEK dataflow and processing

From a user perspective, the system offers a streamlined workflow for geospatial data visualization. After uploading their dataset (supporting formats like CSV, GeoJSON, or Shapefile), users access the data exploration and visualization interface with multiple styling options. The targeted attribute suggestion tab allows users to input natural language prompts that the LLM translates into automatic styling. Alternatively, the automated attribute suggestion tab displays all dataset attributes with explanations, letting users select any attribute for which the system generates appropriate styling code based on its characteristics. All styling changes apply immediately to the visualization, and users can iteratively refine their results by trying different prompts or attributes until they achieve their desired visualization.

Our objective is to enhance visualization capabilities through LLMs and statistical integration. Specifically, this work introduces the following key enhancements:

- **Efficient Statistical Computation:** Precomputed statistical summaries of attributes eliminate need for on-demand computation, reducing query latency and data transmission.
- **Attribute Type Inference:** LLMs automatically identify complex formats (datetime, geometry data, etc.) from small data samples without manual specification.
- **LLM-Driven Attribute Suggestion and Styling Generation:** Automated attribute suggestions and styling code generation based on user prompts, reducing manual effort while ensuring visualization consistency.

2 SYSTEM DESIGN

2.1 Step 1: Attribute Type Inference

The first step of LASEK's visualization pipeline is LLM-driven attribute type inference. Modern data exploration systems often deal with semi-structured data in text-based formats such as CSV and JSON. Some attributes can be very important for visualization but need to be parsed correctly, e.g., date/time and geometry attributes. To address this challenge, LASEK extracts a representative sample of the data and forwards it to the LLM with a prompt to identify datetime and geometry attributes and their format. For datetime attributes, it recognizes various formats such as 'yyyy-mm-dd' or ISO formats. For geometry data, it recognizes common formats such as Well-Known Text (WKT) and coordinate pair representations. The LLM returns a structured JSON response mapping each attribute to its inferred type and format specifics. The returned information can be used to parse the attribute using standard libraries available in JavaScript or other languages. This *enhanced schema* information is stored to enable appropriate parsing logic and visualization techniques without manual configuration.

2.2 Step 2: Statistical Analysis

In order for LLM to suggest attribute styles, it needs to have a holistic look on the dataset. However, its memory is usually very limited and the cost of LLMs increase with the number of tokens. Thus, we need a way to summarize the data into a few tokens to let the LLM understand the big picture. At the same time, this step should be scalable to support large-scale data.

To achieve these goals, we developed a module that pre-computes necessary summaries to eliminate the need of transferring the entire data. To reduce the computation overhead for big data, we leveraged SparkSQL aggregate expressions to execute a single-pass sophisticated query that computes aggregates for all attributes with minimal overhead. We break down the aggregates into *general* aggregates and *specialized* aggregates. General aggregates are computed for all attributes and these include number of non-null values, number of distinct values, top-k values, and their frequencies, which we found to be effective in identifying categorical attributes, e.g., crime type. For efficiency, we approximate the number of distinct values using HyperLogLog (HLL) method. In addition, depending on the attribute type, we compute additional specialized aggregates. For numerical values, we compute the range [min, max], average, standard deviation, and summation, which are useful for deciding gradient color schemes. For textual attributes, we compute the average text length which can help determine if it can be displayed as a label. For datetime attributes, we compute the earliest and latest dates which are helpful for adding range filters. Finally, for geometric attributes, we compute the minimum bounding rectangle (MBR), the distinct geometry types, e.g., points, lines, and polygons, and the total number of points in all geometries. Even though the number of aggregate functions might seem large, we utilize SparkSQL to compute all of them in one pass in a distributed query. We found that the time of computing these summaries for about 50 attributes to be only double the time of calculating the summary for a single geometry column.

2.3 Step 3a: Targeted Attribute Selection

In targeted mode, LASEK identifies the target attribute to visualize and the style options based on a user-provided prompt such as “Show different crime types with distinct colors” for a crime dataset. It works by forwarding the user prompt, the enhanced schema, and a dataset sample to LLM with instructions to identify the most relevant attribute to use in visualization. The output a list of suggestions in the form of (attribute, cue) pairs that can be used for visualization. The cue is a free text description of how the attribute can be used for styling. It is possible that the list contains only one attribute if the prompt very clearly targets a specific attribute. The user can either select one of these options or LASEK can just go with the first option. The selected (attribute, cue) pair is forwarded to the next step.

2.4 Step 3b: Automated Attribute Selection

In automated mode, the system autonomously selects visualization-worthy attributes. This mode can be useful when the dataset contains many attributes and the user is not sure what each of them represents. First, we feed the LLM the enhanced schema and the dataset sample and instruct it to recommend the most relevant attributes for visualization styling. The LLM analyzes the attributes and the sample data and comes up with a list of (attribute, cue) pairs for the most relevant attributes for styling. For example, it can choose the ‘population’ attribute with a cue of “Population is a numeric attribute with significant variance. Use a graduated color scheme.” The list of suggested attributes and their cues is displayed to the user to select the most relevant one. The selected (attribute, cue) pair is then forwarded to the next step.

2.5 Step 4: Style Generation

After an attribute is selected for visualization, whether with targeted or automated mode, the system selects calculated statistical summaries for this specific attribute and feeds it to the LLM through a second call along with the attribute name and cue. While we could send the statistics of all attributes in the first call in Step 3, this two-step approach optimizes the cost of using LLM by minimizing number of transferred tokens and reduces query latency. The LLM then generates a tailored JSON styling object that can be used by Beast and OpenLayers. For example, the style for “Crime Type” that assigns a distinctive color for each type a categorical approach for the “Crime Type" attribute that assigns distinctive colors to different categories (red for theft, blue for battery, etc.). This styling code is applied to the visualization, allowing users to gain insights or iterate with different prompts as needed. We tried to directly generate a styling function in JavaScript and it worked fine but we chose to use a structured styling object for security reasons to avoid directly running the code that the LLM generates.

3 DEMONSTRATION SCENARIOS

This section presents three demonstration scenarios that illustrate the main steps of attribute type inference, attribute recommendation, and styling generation using the LLM-driven approach. For this demonstration, we integrated the Google Gemini LLM into Beast [2]. However, the architecture is designed to be flexible, allowing other LLMs to be seamlessly incorporated. These examples showcase how the system automatically selects relevant attributes based on data samples and user intent, and suggests optimal styling options based on computed statistical summaries. The datasets used for these scenarios are: Chicago crimes and States-and-provinces, downloaded from UCR-Star [4]. We will also have all UCR-Star datasets available for users to further explore LASEK.

3.1 Scenario 1: Visualize Crimes from a Certain Time Period

In this scenario, we will showcase the attribute type inference by LLM, specifically, detecting datetime attributes in the dataset. Users can then use the re-parsed schema to visualize crimes that took place in a certain period.

(1) Attribute Type Inference: In this step, the audience will be able to see the various datetime attributes detected by LLM along with their formats, for example:

```
Date: MM/DD/YY HH:mm
UpdatedOn: M/D/YY HH:mm
Year: YYYY
```

(2) Automated Attribute Selection: LASEK will make the second LLM call to detect the attributes to use for the visualization. Since the schema contains multiple timestamp attributes, the user will be given multiple attribute options to choose from to generate the style.

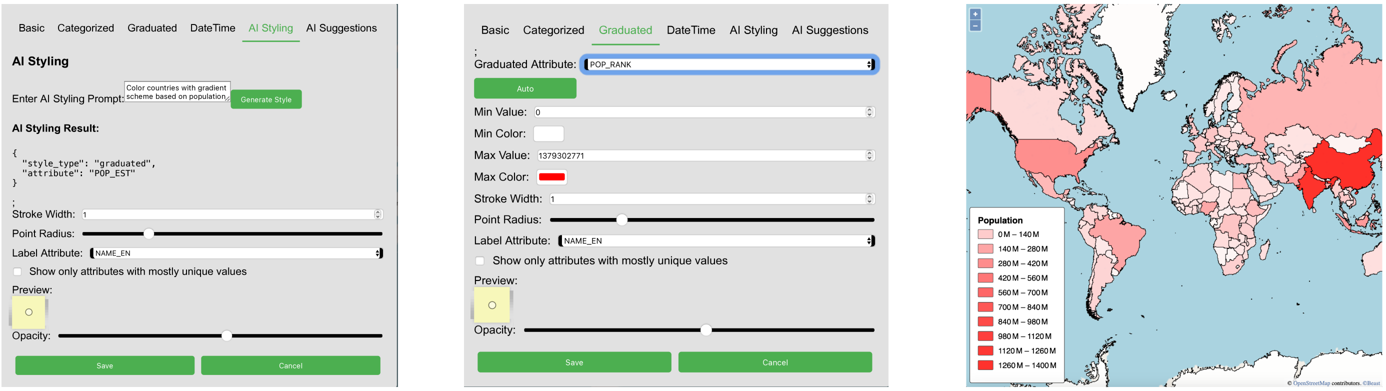
(3) Styling Generation: After the audience makes the selection, the third and final LLM call will take the user choice along with the selected attribute statistics to generate the styling object accordingly. For example, the user may choose to only visualize crimes that took place between 01/01/2022 and 02/10/2022. The generated style will hide all records outside this date range.

3.2 Scenario 2: States and Provinces Data Exploration

This dataset contains 83 attributes, making it challenging for users to identify the most relevant ones. By analyzing both a data sample and the schema, the LLM promptly determines which attributes are best suited for visualization. This data is available in a structured Shapefile format so it does not require attribute type inference.

(1) Attribute Recommendation: In this scenario, we use the automated attribute selection method which identifies several interesting attributes for visualization. For instance, attributes like *mapcolor13* are identified as categorical, while those like *name*, which mostly contain unique values, are more appropriate for labels rather than coloring. Likewise, continuous numerical attributes such as *area_sqkm* are flagged as suitable for a graduated color scheme. For each attribute, the LLM generates an explanation detailing its classification and recommending the optimal styling approach. This explanation is then used as a styling cue in the subsequent generation phase. Users can choose any attribute for which they want to generate styling; for example, they might opt to color regions based on *area_sqkm* while using *name* as labels.

(2) Style Generation: At this point, the LLM combines the selected attribute’s statistical summary with the styling cue to generate a custom styling function, which is then applied to visualize the input data.



Basic Categorized Graduated DateTime AI Styling AI Suggestions

AI Styling

Enter AI Styling Prompt: Color countries with gradient scheme based on population Generate Style

AI Styling Result:

```
{
  "style_type": "graduated",
  "attribute": "POP_EST"
}

;
```

Stroke Width: 1

Point Radius:

Label Attribute: NAME_EN

□ Show only attributes with mostly unique values

Preview:

Opacity:

Save Cancel

(a) Step 1: LASEK sends an initial prompt. LLM receives a sample and the schema information, and recommends best attribute to use and styling type.

Basic Categorized Graduated DateTime AI Styling AI Suggestions

;

Graduated Attribute: POP_RANK

Auto

Min Value: 0

Min Color:

Max Value: 1379302771

Max Color: 🟥

Stroke Width: 1

Point Radius:

Label Attribute: NAME_EN

□ Show only attributes with mostly unique values

Preview:

Opacity:

Save Cancel

(b) Step 2: User selects the recommended attribute. LLM receives statistics of the attribute and generates styling function.

(c) Step 3: Resulting visualization, where countries that have a higher population appear darker.

Figure 2: An example scenario of using LASEK for visualizing countries based on population. The user inputs prompt "Color countries with gradient scheme based on population." The LLM determines which attribute is best to use and generates appropriate styling function.

3.3 Scenario 3: Urban Crime Analysis

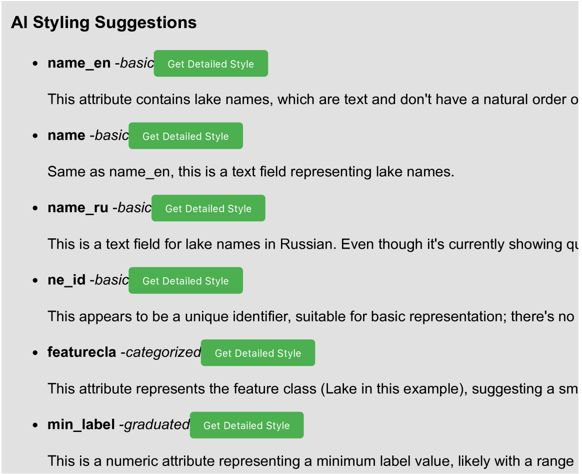
This scenario demonstrates how the audience can refine the style by customizing the visualization prompt.

(1) Targeted Attribute Selection: The audience enters a prompt such as "Show top 5 most frequent crime types with distinct colors". The LLM infers from the schema and sample data that the focus should be on the *Primary Type* attribute.

(2) Style Recommendation: After receiving the summary of *Primary Type* attribute, the system has already detected this attribute as categorical and will create a styling function where the top 5 frequent crime categories appear with distinct colors.

(3) Interactive Customization: A user might later refine the visualization with a natural language prompt such as, "Show the theft incident with larger red circles", prompting the system to adjust the style dynamically.

These scenarios clearly illustrate a two-step process: first, the system intelligently selects relevant attributes based on a data sample and schema, combined with user intent; second, it provides styling recommendations by analyzing the statistical summaries of the selected attributes. This two-step process optimizes number of tokens that are sent to the LLM and reduces query latency. Natural language prompts further enable interactive customization, simplifying the visualization process for non-expert users.



AI Styling Suggestions

- **name_en** -*basic* Get Detailed Style

  This attribute contains lake names, which are text and don't have a natural order o

- **name** -*basic* Get Detailed Style

  Same as name_en, this is a text field representing lake names.

- **name_ru** -*basic* Get Detailed Style

  This is a text field for lake names in Russian. Even though it's currently showing qu

- **ne_id** -*basic* Get Detailed Style

  This appears to be a unique identifier, suitable for basic representation; there's no

- **featurecla** -*categorized* Get Detailed Style

  This attribute represents the feature class (Lake in this example), suggesting a sm

- **min_label** -*graduated* Get Detailed Style

  This is a numeric attribute representing a minimum label value, likely with a range

Figure 3: This figure shows automated attribute recommendations by the LLM for a Lakes dataset.

REFERENCES

[1] 2025. Interactive Maps with OpenLayers. https://openlayers.org/. OpenLayers Documentation.

[2] Ahmed Eldawy et al. 2021. Beast: Scalable Exploratory Analytics on Spatio-temporal Data. In *ACM International Conference on Information and Knowledge Management (CIKM '21)*. 3796–3807.

[3] Saheli Ghosh and Ahmed Eldawy. 2022. AID*: A Spatial Index for Visual Exploration of Geo-Spatial Data. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2022), 3569–3582. https://doi.org/10.1109/TKDE.2020.3026657

[4] Saheli Ghosh, Tin Vu, Mehrad Amin Eskandari, and Ahmed Eldawy. 2019. UCR-STAR: the UCR spatio-temporal active repository. *SIGSPATIAL Special* 11, 2 (Dec. 2019), 34–40. https://doi.org/10.1145/3377000.3377005

[5] Kevin Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César Hidalgo. 2019. VizML: A Machine Learning Approach to Visualization Recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. 1–12. https://doi.org/10.1145/3290605.3300358

[6] Jinfeng Liu, Ziming Liu, Tanja Blascheck, Wencan Fan, and Nan Cao. 2023. Large language models for information visualization: Survey and opportunities. *Visual Informatics* 7, 4 (2023), 113–124.

[7] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Towards a general-purpose query language for visualization recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA'16@SIGMOD)*. Association for Computing Machinery, New York, NY, USA, Article 4, 6 pages.

[8] Yupeng Xie, Yuyu Luo, Guoliang Li, and Nan Tang. 2024. HAIChart: Human and AI Paired Visualization System. *Proc. VLDB Endow.* 17, 11 (July 2024), 3178–3191.

[9] Jia Yu and Mohamed Sarwat. 2021. GeoSparkViz: a cluster computing system for visualizing massive-scale geospatial data. *The VLDB Journal* 30, 2 (Jan. 2021), 22.