Generating Analytic Specifications for Data Visualization from Natural Language Queries using Large Language Models

Subham Sah[π∗]
UNC Charlotte

Rishab Mitra[π†]
Georgia Institute of Technology

Arpit Narechania[π‡]
Georgia Institute of Technology

Alex Endert[§]
Georgia Institute of Technology

John Stasko[¶]
Georgia Institute of Technology

Wenwen Dou[‖]
UNC Charlotte

Abstract

Recently, large language models (LLMs) have shown great promise in translating natural language (NL) queries into visualizations, but their "black-box" nature often limits explainability and debuggability. In response, we present a comprehensive text prompt that, given a tabular dataset and an NL query about the dataset, generates an analytic specification including (detected) data attributes, (inferred) analytic tasks, and (recommended) visualizations. This specification captures key aspects of the query translation process, affording both explainability and debuggability. For instance, it provides mappings from the detected entities to the corresponding phrases in the input query, as well as the specific visual design principles that determined the visualization recommendations. Moreover, unlike prior LLM-based approaches, our prompt supports conversational interaction and ambiguity detection capabilities. In this paper, we detail the iterative process of curating our prompt, present a preliminary performance evaluation using GPT-4, and discuss the strengths and limitations of LLMs at various stages of query translation. The prompt is open-source and integrated into NL4DV, a popular Python-based natural language toolkit for visualization, which can be accessed at **https://nl4dv.github.io**.

Index Terms: Large language models; Natural language interfaces; Visualization; Prompt engineering;

1 Introduction and Background

Data visualization is an important component of data-driven storytelling [17]. However, existing tools for creating visualizations often require specialized knowledge, either through programming or using a graphical user interface (GUI), which limits authoring, customization, and analysis capabilities to experts.

One way to overcome this limitation and increase user access is by using natural language (NL) to create visualizations. Cox et al. [7] first introduced the concept of creating visualizations from structured NL commands (NL2VIS). Since then, many natural language interfaces (NLIs) and toolkits for visualization have emerged that use keyword-based or semantic parsing-based approaches to interpret queries [11, 43, 29, 26, 16, 36, 35, 37, 39]. NL4DV [29, 26] is one such toolkit that utilizes a rules-based approach, employing dependency parsers like CoreNLP [24] to provide an analytic specification that includes detected attributes, inferred tasks, and visualization recommendations from an NL query and dataset. However, approaches like NL4DV require developers to create complex rules, which can limit the range and flexibility of input NL queries. Advancements in natural language processing (NLP) and deep learning have further improved NL2VIS systems, which utilize transformers to interpret queries [21, 20].

More recently, large language models (LLMs) like GPT-4 [33], Claude [3], and Gemini [12] have been shown to effectively analyze and extract meaningful information, key concepts, relationships, and trends from unstructured textual data [25]. These capabilities have since been utilized for creative writing [13], code generation [5, 15], dataset curation [18], and visualization creation [6, 34, 9, 41]. One notable LLM-based visualization system, chartGPT [41], has outperformed a parsing-based system (NL4DV [29]) and a deep-learning based system (ncNet [21]). In spite of their superior performance, LLM-based systems have certain documented limitations, such as providing insufficient explanations for the system's generated output [8] and being inconsistent in generating visualizations [23]. These unexplainable, uncertain systems impact transparency and trust, making it difficult for users to find and fix errors. In the NL to SQL domain, several explainable systems have already helped users identify and fix errors in the generated SQL queries [30, 10, 27], motivating this work for more explainable NL2VIS scenarios.

In this work, we present a new LLM-based text prompt (NL4DV-LLM) that, like NL4DV [29, 26], returns an analytic specification containing data attributes, analytic tasks, and relevant visualizations. This specification, presented as a structured JSON object (preferred by developers) or a step-by-step natural language explanation (preferred by users), affords explainability and debuggability by documenting key aspects of the query translation process. For example, it provides the mappings between the detected entities and the corresponding phrases in the input query as well as the specific visual design principles that determined the visualization recommendation. Furthermore, this prompt offers conversational interaction and ambiguity detection functionalities which are currently unsupported in other LLM-based NL2VIS systems. Essentially, users can ask follow-up queries to alter previously generated analytic specification(s) based on their evolving needs. If a query also contains ambiguities (e.g. a query phrase that can map to multiple data attributes), the prompt outputs multiple visualizations, one for each ambiguous entity. However, the prompt does not support query resolution since it is a programmatic capability outside the scope of query translation [26]. Figure 1 highlights this key difference in the capabilities of NL4DV and NL4DV-LLM.

We also conducted a preliminary evaluation of NL4DV-LLM against NL4DV using the NLVCorpus dataset [38] and GPT-4 [33] as the prompt's engine. We found that in our corpus of 740 queries across three datasets, our prompt achieved an accuracy of 87.02% compared to NL4DV's accuracy of 64.05%. However, on average, the LLM took around 25 seconds to generate analytic specifications, which can be a potentially unreasonable wait time for users. We discuss the tradeoffs and strengths of this prompt, which we hope guides future developments in natural language to visualiza-

[∗]e-mail: ssah1@uncc.edu
[†]e-mail: rmitra34@gatech.edu
[‡]e-mail: arpitnarechania@gatech.edu
[§]e-mail: endert@gatech.edu
[¶]e-mail: stasko@cc.gatech.edu
[‖]e-mail: wdou1@uncc.edu
[π]authors contributed equally

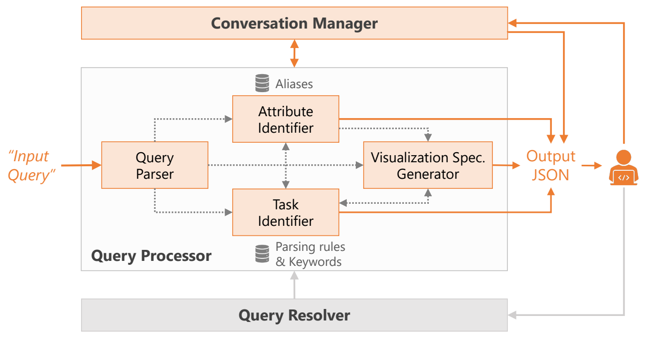


Figure 1: Architecture diagram of NL4DV [29, 26] highlighting the modules supported by NL4DV-LLM (in orange), with arrows indicating the flow of information.

tion interfaces. Our primary contributions are the following:

- An LLM-based NL2VIS text prompt (NL4DV-LLM) that translates NL queries about a tabular dataset into a comprehensive analytic specification that includes detected attributes, analytic tasks, and a recommended visualization.
- Description of our iterative process to curate the prompt.
- Findings from a preliminary evaluation of NL4DV-LLM against NL4DV and a discussion on the prompt's strengths and tradeoffs.

2 Developing the NL4DV-LLM Prompt

To make LLM-based NL2VIS systems more explainable, we curated a text prompt, NL4DV-LLM. Given a tabular dataset and a natural language query about the dataset, this prompt produces a detailed analytic specification, including data attributes, analytic tasks, visualizations, and additional metadata explaining the translation process. NL4DV-LLM's analytic specification essentially tries to replicate the output of NL4DV [29], a popular semantic parsing-based toolkit. In this section, we describe the components of our prompt and the iterative process of engineering it.

2.1 Prompt Components

Figure 2 illustrates the various components of NL4DV-LLM along with several example visualization outputs on different kinds of input queries. We describe each component below.

2.1.1 Analytic Tasks & Visualization Design Knowledge

NL4DV's output analytic specification includes Amar et al.'s [1] low-level components of analytic activity (i.e. "analytic tasks") as inferred from the input query. To include these analytic tasks as part of NL4DV-LLM's output, we first probed GPT-4 to check if it has 'learnt' the theory about these analytic tasks during its training. Upon finding GPT-4 has *not yet* learnt about analytic tasks, we decided to supply this knowledge in the prompt. Specifically, we curated a structured JSON comprising the **Name**, **Description**, a **Pro Forma Abstract**, **Examples**, **Attribute Data Types and Visual Encodings**, **Attributes and Visual Encodings Description**, and **Recommended Visualization** for seven analytic tasks (*Correlation, Distribution, Derived Value, Trend, Filter, Sort, and Find Extremum*). The **Attribute Data Types and Visual Encodings** property details the preferred visual encodings (e.g., "X axis") and the datatypes (e.g., "Quantitative") that can be mapped to them; the **Attributes and Visual Encodings Description** provides instructions on how to encode these visual encodings and datatypes in Vega-Lite. The **Recommended Visualization** property specifies the visualization types most suitable for the given task, according to the design heuristics in NL4DV [29].

Using an "in-context-learning" [19] approach, we include this structured **Task JSON** as part of the main prompt. We format this taxonomy as a JSON for succinctness and clarity, and to remove any potential ambiguities that may arise when utilizing NL.

2.1.2 Conversational Interaction

Similar to NL4DV [29], NL4DV-LLM supports modifications to previously generated visualizations via follow-up queries. NL4DV utilizes a unique follow-up taxonomy that classifies follow-up queries as one of three types: add, remove, replace for one or more components (specifically attributes, tasks, visualization types) of an analytic specification. We replicate this taxonomy through a JSON array, shown in Figure 2, that contains its resultant permutations (e.g. add + analytic task), instructions describing the necessary steps to perform each operation, and follow-up query examples to provide context for each permutation.

2.1.3 Instructions Based on Query Type

The key instruction in our prompt to output an analytic specification is as follows:

"...classify the below natural language queries into the respective analytic tasks they map to. There can be one or more analytic tasks detected in the input natural language query. Return the visualization type in the form of a Vega-Lite specification where it reads data from the url above."

The instructions included in this component specify explicit steps on handling common NL query types: ambiguous queries, fully specified queries, underspecified queries, and follow-up queries. Figure 2 illustrates an example of each query type and its corresponding visualization.

Underspecified Queries: We utilize NL4DV's concept of underspecified queries, which is defined as queries that "*implicitly refer to tasks and visualizations* [29]. If the query does not contain explicit references to tasks or visualizations, then the prompt instructs the LLM to utilize the design guidelines posited by the Analytic Task JSON, "*infer the task that is best suited with the detected attributes' datatypes*", and "*generate a visualization specification using this inferred task and detected attributes*".

Fully Specified Queries: We also define fully specified queries like NL4DV, where the NL query makes explicit references to at least one attribute, task, and visualization type [29]. The key instruction described previously is sufficient to ensure coverage of fully specified queries; no other instructions are required.

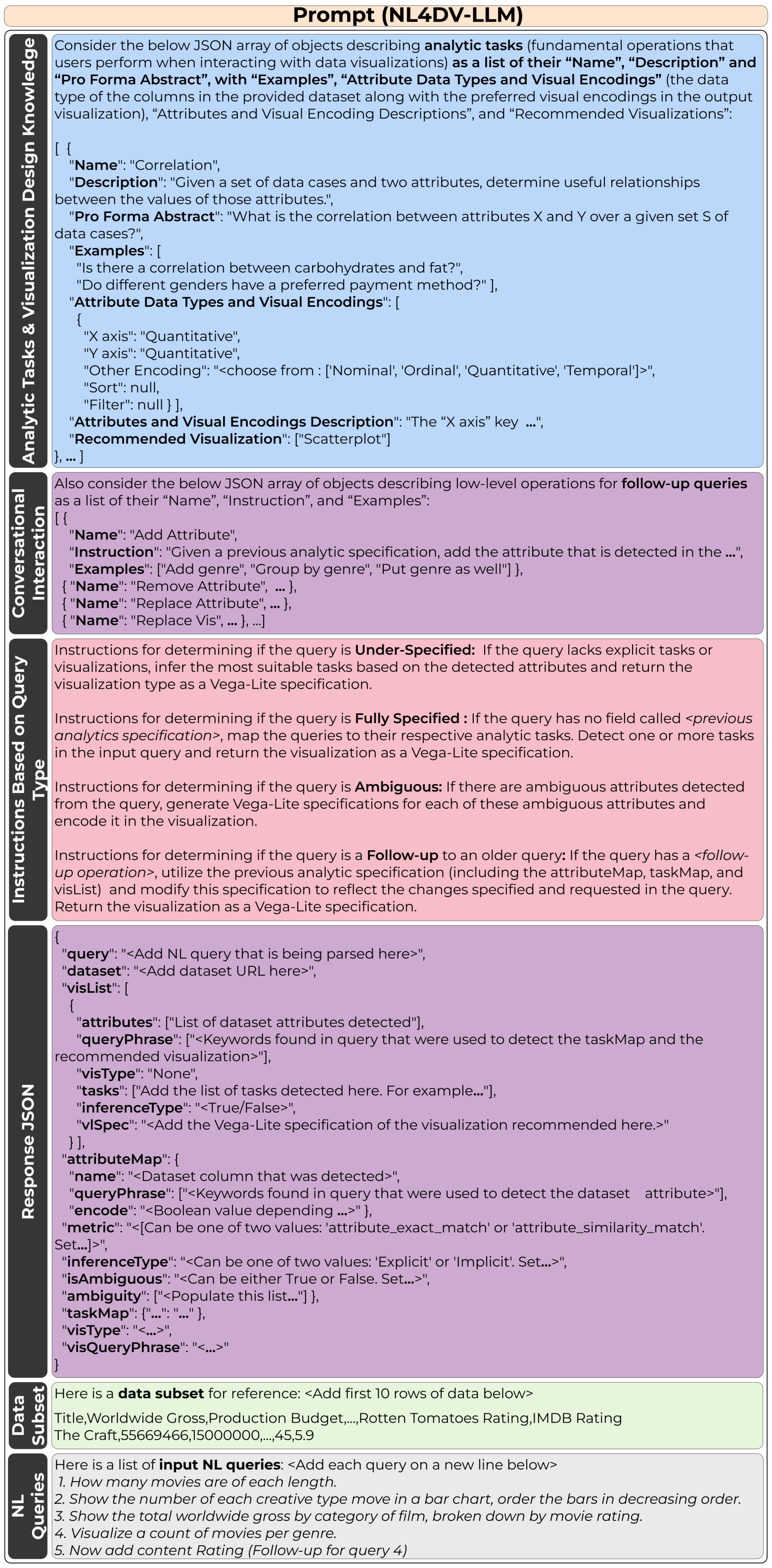
Ambiguous Queries: The prompt defines ambiguous queries as queries "*with partial references to multiple data attributes.*" In such cases, the prompt instructs outputting multiple visualizations, one for every attribute that a keyword potentially refers to, to maximally cover the user's intent.

Follow-up Queries: Since follow-up queries alter components (tasks, attributes, or visualization types) of a previously generated analytic specification rather than creating an entirely new one, the prompt includes another set of instructions to handle follow-up queries. For this query type, users must append a previously generated analytic specification to the end of the prompt. With this previously generated analytic specification, the primary instruction to handle follow-up queries is as follows:

"...classify the below natural language query into the respective follow-up operations they map to. Utilize the previous analytic specification (including the attributeMap, taskMap, and visList) and modify this specification to reflect the changes specified and requested in the natural language query. Return the visualization type in the form of a Vega-Lite specification where it reads data from the url above."

2.1.4 Response JSON

As shown in Figure 2, NL4DV-LLM's output replicates NL4DV's, and contains an *attributeMap* that is composed of the dataset attributes inferred from the natural language query, a *taskMap* com-



Prompt (NL4DV-LLM)

Analytic Tasks & Visualization Design Knowledge

Consider the below JSON array of objects describing **analytic tasks** (fundamental operations that users perform when interacting with data visualizations) **as a list of their "Name", "Description" and "Pro Forma Abstract", with "Examples", "Attribute Data Types and Visual Encodings"** (the data type of the columns in the provided dataset along with the preferred visual encodings in the output visualization), "Attributes and Visual Encoding Descriptions", and "Recommended Visualizations":

```
[ {
   "Name": "Correlation",
   "Description": "Given a set of data cases and two attributes, determine useful relationships
between the values of those attributes.",
   "Pro Forma Abstract": "What is the correlation between attributes X and Y over a given set S of
data cases?",
   "Examples": [
     "Is there a correlation between carbohydrates and fat?",
     "Do different genders have a preferred payment method?" ],
   "Attribute Data Types and Visual Encodings": [
     {
       "X axis": "Quantitative",
       "Y axis": "Quantitative",
       "Other Encoding": "<choose from : ['Nominal', 'Ordinal', 'Quantitative', 'Temporal']>",
       "Sort": null,
       "Filter": null } ],
   "Attributes and Visual Encodings Description": "The "X axis" key  ...",
   "Recommended Visualization": ["Scatterplot"]
}, ... ]
```

Conversational Interaction

Also consider the below JSON array of objects describing low-level operations for **follow-up queries** as a list of their "Name", "Instruction", and "Examples":

```
[ {
   "Name": "Add Attribute",
   "Instruction": "Given a previous analytic specification, add the attribute that is detected in the ...",
   "Examples": ["Add genre", "Group by genre", "Put genre as well"] },
  { "Name": "Remove Attribute",  ... },
  { "Name": "Replace Attribute", ... },
  { "Name": "Replace Vis", ... }, ...]
```

Instructions Based on Query Type

Instructions for determining if the query is **Under-Specified:** If the query lacks explicit tasks or visualizations, infer the most suitable tasks based on the detected attributes and return the visualization type as a Vega-Lite specification.

Instructions for determining if the query is **Fully Specified :** If the query has no field called *<previous analytics specification>*, map the queries to their respective analytic tasks. Detect one or more tasks in the input query and return the visualization as a Vega-Lite specification.

Instructions for determining if the query is **Ambiguous:** If there are ambiguous attributes detected from the query, generate Vega-Lite specifications for each of these ambiguous attributes and encode it in the visualization.

Instructions for determining if the query is a **Follow-up** to an older query**:** If the query has a *<follow-up operation>*, utilize the previous analytic specification (including the attributeMap, taskMap, and visList) and modify this specification to reflect the changes specified and requested in the query. Return the visualization as a Vega-Lite specification.

Response JSON

```
{
  "query": "<Add NL query that is being parsed here>",
  "dataset": "<Add dataset URL here>",
  "visList": [
    {
      "attributes": ["List of dataset attributes detected"],
      "queryPhrase": ["<Keywords found in query that were used to detect the taskMap and the
recommended visualization>"],
      "visType": "None",
      "tasks": ["Add the list of tasks detected here. For example..."],
      "inferenceType": "<True/False>",
      "vlSpec": "<Add the Vega-Lite specification of the visualization recommended here.>"
    } ],
  "attributeMap": {
    "name": "<Dataset column that was detected>",
    "queryPhrase": ["<Keywords found in query that were used to detect the dataset    attribute>"],
    "encode": "<Boolean value depending ...>" },
  "metric": "<[Can be one of two values: 'attribute_exact_match' or 'attribute_similarity_match'.
Set...]>",
  "inferenceType": "<Can be one of two values: 'Explicit' or 'Implicit'. Set...>",
  "isAmbiguous": "<Can be either True or False. Set...>",
  "ambiguity": ["<Populate this list..."] },
  "taskMap": {"...": "..." },
  "visType": "<...>",
  "visQueryPhrase": "<...>"
}
```

Data Subset

Here is a **data subset** for reference: <Add first 10 rows of data below>

Title,Worldwide Gross,Production Budget,...,Rotten Tomatoes Rating,IMDB Rating
The Craft,55669466,15000000,...,45,5.9

NL Queries

Here is a list of **input NL queries**: <Add each query on a new line below>
1. How many movies are of each length.
2. Show the number of each creative type move in a bar chart, order the bars in decreasing order.
3. Show the total worldwide gross by category of film, broken down by movie rating.
4. Visualize a count of movies per genre.
5. Now add content Rating (Follow-up for query 4)

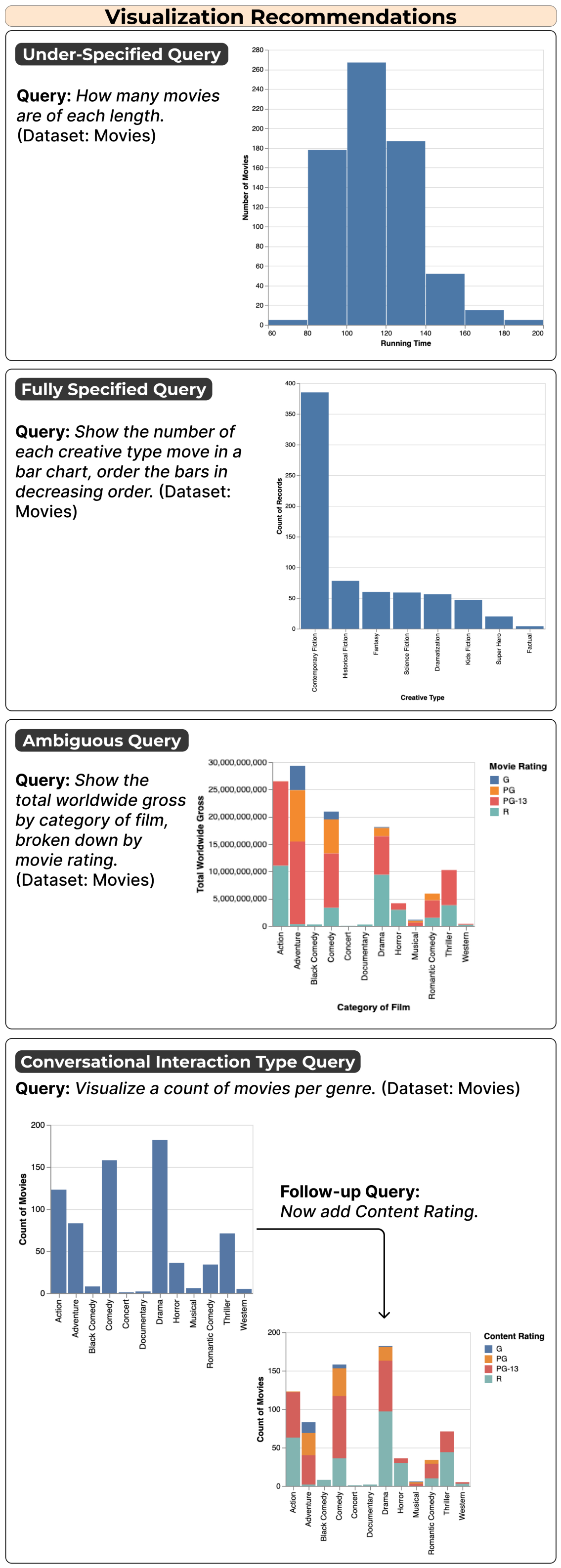


Figure 2: NL4DV-LLM prompt for Visualization Generation

posed of the inferred analytic tasks, and a *visList* that includes the Vega-Lite specifications relevant to the query [29]. An example of this response is provided in the prompt itself, along with the instruction, "*Here is the JSON object that the response should be returned as*". Also shown in Figure 2, the example JSON object includes explicit instructions for each property, which constrains the output as much as possible, thereby increasing the prompt's consistency.

We provide an additional constraint in NL4DV-LLM's prompt by stating, *"Do not include any additional prose in your response. I only want to see the JSON."* This instruction ensures that the LLM only outputs the response JSON object, suitable for developers who are building NL2VIS applications on top of the prompt. Without this instruction, the LLM provides natural language explanations for its steps in formulating the response JSON for a given NL query. Users can opt to delete this sentence from the prompt if they would like to view additional explainable behavior from the LLM.

2.1.5 Data Subset

For the LLM to detect references to attributes and records in the input query, it needs access to the entire dataset. However, due to a token limit imposed by the GPT-4 API, it is not possible to include the entire dataset as part of the prompt. Consequently, we select all dataset headers (columns) and randomly subset ten records (rows) for the prompt. While this choice alleviates the token-limit concern, it still has two limitations. First, for really wide datasets, the large number of columns may still exceed the token limit; and second, the ten sample rows may not be representative of the entire dataset and may result in false negatives in the query translating process. To resolve cases where the amount of dataset columns exceeds the prompt's token limit, users can leverage methods in exploratory data analysis such as Linear Discriminant Analysis [40] that only includes the most relevant columns as part of the prompt.

2.1.6 NL Queries

We finally conclude our prompt by including a list of one or more input natural language queries for processing.

2.2 Prompt Iterations

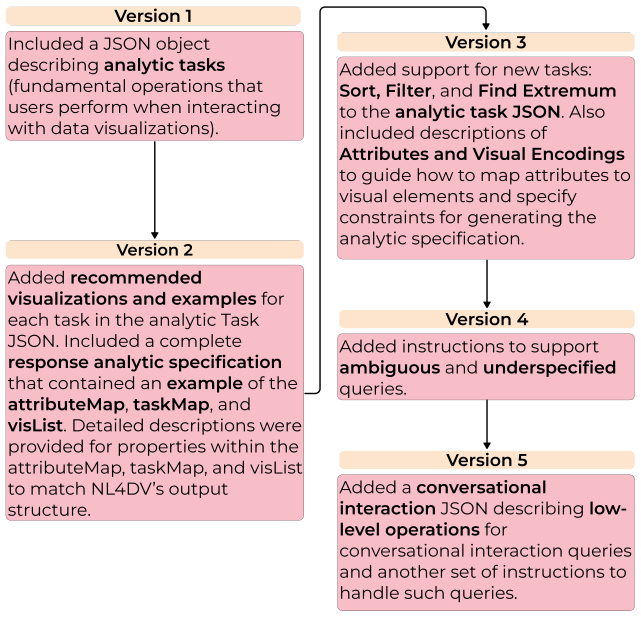

Version 1
Included a JSON object describing **analytic tasks** (fundamental operations that users perform when interacting with data visualizations).
Version 2
Added **recommended visualizations and examples** for each task in the analytic Task JSON. Included a complete **response analytic specification** that contained an **example** of the **attributeMap**, **taskMap**, and **visList**. Detailed descriptions were provided for properties within the attributeMap, taskMap, and visList to match NL4DV's output structure.
Version 3
Added support for new tasks: **Sort, Filter**, and **Find Extremum** to the **analytic task JSON**. Also included descriptions of **Attributes and Visual Encodings** to guide how to map attributes to visual elements and specify constraints for generating the analytic specification.
Version 4
Added instructions to support **ambiguous** and **underspecified** queries.
Version 5
Added a **conversational interaction** JSON describing **low-level operations** for conversational interaction queries and another set of instructions to handle such queries.

Figure 3: Illustration of our iterative prompt development process.

Figure 3 illustrates how our prompt underwent multiple iterations to improve its performance in analytic specification generation and support a wide variety of NL queries. The initial version of the prompt contained a JSON object describing a subset of analytic tasks found in Amar et al. [1]: *Correlation*, *Derived Value*, *Distribution*, and *Trend*. However, our prompt was often unable to detect the correct tasks for a given NL query. In addition, the prompt only outputted a Vega-Lite specification with no explanations whatsoever, affording limited explainability to the user. In Version 2, we augmented each task in the analytics task JSON object with example queries and recommended visualizations. In addition, we also included a sample analytic specification, similar to NL4DV's [29], including an *attributeMap* and a *taskMap*, to enhance the prompt's explainability to the user. For example, users could now view the mappings between the detected dataset attributes and the corresponding phrase in the input NL query.

Next, in Version 3, we enhanced the analytic task JSON by introducing support for other analytic tasks, namely *Sort, Filter, and Find Extremum*. An *Attributes and Visual Encodings Description* property was also included for each task, providing the LLM heuristics on task inference for underspecified queries. Version 4 provided additional instructions to handle ambiguous and underspecified queries since earlier prompt versions were unable to generate accurate visualizations for such query types.

Finally, in version 5, we introduced a conversational interaction JSON describing the low-level taxonomy introduced in Mitra et al. [26] and a distinct set of instructions to process follow-up queries. With this version, we ensure coverage of all natural language query types supported by NL4DV [29].

3 Preliminary Evaluation

3.1 Setup and Design

To study our prompt's performance across different NL2VIS scenarios, we conducted a preliminary evaluation using GPT-4 as our prompt engine and NLVCorpus [38] as our dataset benchmark (query corpus). We chose GPT-4 as it was considered as the state-of-the-art LLM during the time of evaluation (February - March 2024). We chose NLVCorpus for its human-generated utterance sets, which provide a realistic and robust representation of NL queries across three dataset domains: movies, cars, and superstore.

To evaluate our prompt's conversational interaction capabilities, we included follow-up queries from the NL4DV website [31]. We did not use NLVCorpus' sequential queries, as many requested unsupported aesthetic changes (e.g., *Use major gridlines*). Our composite dataset comprised 740 queries. We executed all 740 queries via our prompt on GPT-4 and evaluated the outputs for correctness. We defined a query to be correct if the generated analytic specification accurately captured the query intent and the resultant visualization included everything asked for in the query.

We provide the dataset corpus (used for evaluation), text prompt (NL4DV-LLM), annotations on the prompt's outputs, and a gallery of sample visualization outputs which can be accessed at `https://github.com/nl4dv/NL4DV-LLM-supplemental-material`.

3.2 Data Annotation Procedure

We process all the queries in our evaluation corpus through NL4DV and NL4DV-LLM and record the response times for each query. We assess whether the output for each query should be deemed as "accurate," signifying if the visualization matched the query's intent and included all analytic tasks and attributes requested by the query. If the output failed to meet these requirements, we marked the visualization as "inaccurate" and specified factors that caused the visualization's inaccuracy. Common reasons for inaccuracies were that the visualization was missing an analytic task, attribute, or encoded incorrect attribute(s). Furthermore, the systems sometimes did not generate outputs for certain queries. The outputs, in these cases, were marked as wholly inaccurate.

The first two co-authors served as annotators to evaluate the accuracy of the resulting visualizations. They first analyzed small subsets of the dataset to ensure that their analyses were fully calibrated with each other, creating a consistent standard in their analysis. Then they went through the systems' responses for the entire corpus of queries and determined if the responses precisely answered the queries. They compared their annotations with each other and discussed any discrepancies in their results to come to a consensus. In cases where the first two co-authors remained split in their decision, the third co-author would use their decision as the tiebreaker for the analysis.

Since many of the input queries in the evaluation dataset could be considered as underspecified or ambiguous, multiple visualizations can be regarded as "accurate" for these types of queries. For example, in the movies' dataset, there can be multiple viable visualizations for the query "*Correlate budget, gross, and rating*", where the keyword "*rating*" can refer to the attributes "*Content Rating*", "*Rotten Tomatoes Rating*", or "*IMDb Rating*". For such cases, the annotators assessed if the output contained any valid interpretation of the query. Therefore, the annotators opted not to refer to the ground truth provided for each query in NLVCorpus for the majority of their annotations, and instead only referred to the ground truth for any discrepancies in their results.

3.3 Results and Discussion

By manually analyzing and annotating every visualization generated by each NL4DV-LLM & NL4DV, we discover that NL4DV-LLM with GPT-4 outperforms the NL4DV in accuracy but has significantly longer response times.

Out of the 740 queries, NL4DV-LLM generated 644 accurate responses, resulting in an accuracy rate of 87.02%. NL4DV generated 474 accurate responses, resulting in an accuracy rate of 64.05%. The prompt's accuracy was mostly consistent across the three test datasets: 84.98% on the cars' dataset, 89.89% accuracy on the movies' dataset, and 86.5% on the superstore dataset. However, NL4DV's accuracies on the movies' dataset (75.49%) and cars dataset (70.73%) were significantly higher than the superstore dataset (40%), potentially due to the superstore dataset's complexity and high syntactic similarity among its attributes. A previous evaluation reported that learning-based NL2VIS system ncNet [21] had an accuracy of about 45% and another LLM-based system chartGPT [41] had an accuracy of about 79%, albeit on the nvBench dataset and a subtle distinction in its definition of accuracy [41]. These findings suggest that our prompt's accuracy is comparable to, if not slightly higher than, previous systems.

Notably, our prompt handled a wide range of query structures and forms, including underspecified and ambiguous queries. For example, our prompt was able to apply transformations and computations to dataset values to create new "derived" attributes (e.g. creating a new attribute *Profit* from *Production Budget* and *Worldwide Gross* in the movies dataset), which semantic parsing-based approaches cannot generally support. However, there were a few queries that resulted in inaccurate outputs. Common reasons for inaccuracy were malformed response JSON objects or well-formed JSON objects with incorrect Vega-Lite syntax. Such occurrences can undermine user trust, highlighting the need for future systems to implement appropriate safeguards. In addition, certain outputs were misleading due to incorrect associations between the data attributes and the visual encodings. For example, for the query, "*Show total profit across genres,*" the y-axis might be labeled "Total profit" but actually use "Worldwide Gross" as the data field, making the visualization confusing to the user.

Lastly, NL4DV-LLM's average response time across all 740 queries was 25 seconds, significantly longer than NL4DV's average response time of 3 seconds [29]. Such a long wait time can impact the prompt's usability. We attribute this high response time to our prompt's extensive size. However, initial testing with another LLM (GPT-4o mini) demonstrated improved response times.

4 Limitations and Future Work

We aimed to make our prompt as explainable as possible by modeling the query interpretation process into the analytic specification JSON. This approach helped us explain system behavior across different datasets and queries. However, in the prompt, we also looked to incorporate confidence scores for the detected attributes, tasks, and visualization types in the response JSON, much like the confidence scores in NL4DV [29]. The confidence scores in NL4DV measure the semantic and syntactic similarity between the detected entities and the query phrases that they map to through functions like cosine similarity and Wu-Palmer scoring [42]. However, whilst implementing this feature, we found that GPT-4 was unable to properly calculate token similarity scores or produce confidence scores meaningful to the user. This shows a limitation in our prompt's explainability absent in parsing-based NL2VIS systems.

Next, we conducted our preliminary evaluation during February 2024 and March 2024. At the time, OpenAI's GPT-4 was considered as the state-of-the-art LLM. Since then, there have been a number of other LLMs released, like Claude-3.5 [4] and GPT-4o [32] that have reported outperforming GPT-4 in a number of benchmarking tasks. Furthermore, GPT-4 itself may have undergone updates and changes, potentially affecting consistency, output accuracy, and response time. Future work is planned to conduct evaluations across different LLMs.

Finally, we conducted our preliminary evaluation using the NLVCorpus dataset, which only contains queries for three datasets. For a more thorough evaluation, we plan to utilize more diverse and comprehensive dataset benchmarks such as nvBench [22], which contains over 25,000 queries for 105 datasets.

5 Conclusion

We presented NL4DV-LLM, an LLM-based prompt that generates analytic specifications from a dataset subset and a natural language query, supporting traditional NL2VIS capabilities. Preliminary evaluation shows promise but also highlights limitations affecting trust and usability. We share our prompt curation experiences to guide future developers in NL2VIS tasks.

Acknowledgments

This material is based upon work supported by NSF CNS-2323795. We thank members of the Georgia Tech Visualization Lab and the Charlotte Visualization Center for their helpful feedback at different stages of this work. We used Google Scholar [14], vitaLITy [28], and vitaLITy 2 [2] to assist us during our literature review.

References

[1] R. A. Amar, J. R. Eagan, and J. T. Stasko. Low-level components of analytic activity in information visualization. *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pp. 111–117, 2005. 2, 4

[2] H. An, A. Narechania, E. Wall, and K. Xu. vitaLITy 2: Reviewing Academic Literature Using Large Language Models. *NLVIZ Workshop (IEEE VIS)*, 2024. 5

[3] Anthropic. Model card and evaluations for claude models. 2023. 1

[4] Anthropic. Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet, Year accessed. Accessed: 2024-07-19. 5

[5] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 1

[6] Q. Chen, S. Pailoor, C. Barnaby, A. Criswell, C. Wang, G. Durrett, and I. Dillig. Type-directed synthesis of visualizations from natural language queries. *Proc. ACM Program. Lang.*, 6(OOPSLA2), oct 2022. doi: 10.1145/3563307 1

[7] K. Cox, R. E. Grinter, S. L. Hibino, L. J. Jagadeesan, and D. Mantilla. A multi-modal natural language interface to an information visualization environment. *International Journal of Speech Technology*, 4:297–314, 2001. 1

[8] V. Dibia. LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models". In D. Bollegala, R. Huang, and A. Ritter, eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 113–126. Association for Computational Linguistics, Toronto, Canada, July 2023. doi: 10.18653/v1/2023.acl-demo.11 1

[9] V. Dibia and Çağatay Demiralp. Data2Vis: Automatic Generation of Data Visualizations Using Sequence to Sequence Recurrent Neural Networks, 2018. 1

[10] A. Elgohary, C. Meek, M. Richardson, A. Fourney, G. Ramos, and A. H. Awadallah. NL-EDIT: Correcting semantic parse errors through natural language interaction. *arXiv preprint arXiv:2103.14540*, 2021. 1

[11] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th annual acm symposium on user interface software & technology*, pp. 489–500, 2015. 1

[12] Gemini. https://gemini.google.com/. Accessed: July 18, 2024. 1

[13] C. Gómez-Rodríguez and P. Williams. A confederacy of models: A comprehensive evaluation of LLMs on creative writing. *arXiv preprint arXiv:2310.08433*, 2023. 1

[14] Google Scholar. https://scholar.google.com, 2024. 5

[15] Z. Gu, J. Fan, N. Tang, S. Zhang, Y. Zhang, Z. Chen, L. Cao, G. Li, S. Madden, and X. Du. Interleaving Pre-Trained Language Models and Large Language Models for Zero-Shot NL2SQL Generation. *arXiv preprint*, 2023. 1

[16] J.-F. Kassel and M. Rohs. Valletto: A multimodal interface for ubiquitous visual analytics. In *Extended abstracts of the 2018 CHI conference on human factors in computing systems*, pp. 1–6, 2018. 1

[17] C. N. Knaflic. *Choosing an Effective Visual*, chap. 2, pp. 35–69. John Wiley & Sons, Ltd, 2015. doi: 10.1002/9781119055259.ch2 1

[18] H.-K. Ko, H. Jeon, G. Park, D. H. Kim, N. W. Kim, J. Kim, and J. Seo. Natural language dataset generation framework for visualizations powered by large language models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–22, 2024. 1

[19] G. Li, X. Wang, G. Aodeng, S. Zheng, Y. Zhang, C. Ou, S. Wang, and C. H. Liu. Visualization Generation with Large Language Models: An Evaluation. *arXiv preprint arXiv:2401.11255*, 2024. 2

[20] C. Liu, Y. Han, R. Jiang, and X. Yuan. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pp. 11–20. IEEE, 2021. 1

[21] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin. Natural Language to Visualization by Neural Machine Translation. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2021. doi: 10.1109/TVCG.2021.3114848 1, 5

[22] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226, 2021. 5

[23] P. Maddigan and T. Susnjak. Chat2VIS: Generating Data Visualizations via Natural Language Using ChatGPT, Codex and GPT-3 Large Language Models. *IEEE Access*, 11:45181–45193, 2023. doi: 10.1109/ACCESS.2023.3274199 1

[24] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In K. Bontcheva and J. Zhu, eds., *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60. Association for Computational Linguistics, Baltimore, Maryland, June 2014. doi: 10.3115/v1/P14-5010 1

[25] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao. Large Language Models: A Survey, 2024. 1

[26] R. Mitra, A. Narechania, A. Endert, and J. Stasko. Facilitating conversational interaction in natural language interfaces for visualization. In *2022 IEEE Visualization and Visual Analytics (VIS)*, pp. 6–10. IEEE, 2022. 1, 2, 4

[27] A. Narechania, A. Fourney, B. Lee, and G. Ramos. DIY: Assessing the correctness of natural language to SQL systems. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*, pp. 597–607, 2021. 1

[28] A. Narechania, A. Karduni, R. Wesslen, and E. Wall. vitaLITy: Promoting Serendipitous Discovery of Academic Literature with Transformers & Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):486–496, 2022. doi: 10.1109/TVCG.2021.3114820 5

[29] A. Narechania, A. Srinivasan, and J. Stasko. NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, 2020. 1, 2, 4, 5

[30] Z. Ning, Y. Tian, Z. Zhang, T. Zhang, and T. J.-J. Li. Insights into Natural Language Database Query Errors: From Attention Misalignment to User Handling Strategies. *ACM Transactions on Interactive Intelligent Systems*, 2024. 1

[31] Sample Queries: NL4DV Showcase. https://nl4dv.github.io/nl4dv/showcase.html. Accessed: July 18, 2024. 4

[32] OpenAI. GPT-4o Contributions. https://openai.com/gpt-4o-contributions/, 2024. Accessed: 2024-07-19. 5

[33] OpenAI et al. GPT-4 Technical Report, 2024. 1

[34] G. Poesia, O. Polozov, V. Le, A. Tiwari, G. Soares, C. Meek, and S. Gulwani. Synchromesh: Reliable code generation from pre-trained language models, 2022. 1

[35] V. Setlur, M. Tory, and A. Djalali. Inferencing underspecified natural language utterances in visual analysis. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, p. 40–51. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3301275.3302270 1

[36] A. Srinivasan, B. Lee, N. Henry Riche, S. M. Drucker, and K. Hinckley. InChorus: Designing consistent multimodal interactions for data visualization on tablet devices. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pp. 1–13, 2020. 1

[37] A. Srinivasan, B. Lee, and J. Stasko. Interweaving Multimodal Interaction With Flexible Unit Visualizations for Data Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 27(8):3519–3533, Aug. 2021. doi: 10.1109/tvcg.2020.2978050 1

[38] A. Srinivasan, N. Nyapathy, B. Lee, S. M. Drucker, and J. Stasko. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–10, 2021. 1, 4

[39] A. Srinivasan and J. Stasko. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):511–521, 2018. doi: 10.1109/TVCG.2017.2745219 1

[40] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien. Linear discriminant analysis: A detailed tutorial. *AI Commun.*, 30(2):169–190, jan 2017. doi: 10.3233/AIC-170729 4

[41] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu. ChartGPT: Leveraging LLMs to Generate Charts from Abstract Natural Language, 2023. 1, 5

[42] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *ACL '94: Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, p. 133–138. Association for Computational Linguistics, USA, 1994. doi: 10.3115/981732.981751 5

[43] B. Yu and C. T. Silva. FlowSense: A natural language interface for visual data exploration within a dataflow system. *IEEE transactions on visualization and computer graphics*, 26(1):1–11, 2019. 1