nvBench 2.0: A Benchmark for Natural Language to Visualization under Ambiguity [Experiment, Analysis & Benchmark]

Tianqi Luo
HKUST(GZ)

Chuhan Huang
HKUST(GZ)

Leixian Shen
HKUST

Boyan Li
HKUST(GZ)

Shuyu Shen
HKUST(GZ)

Wei Zeng
HKUST(GZ)

Nan Tang
HKUST(GZ)

Yuyu Luo
HKUST(GZ)

ABSTRACT

Natural Language to Visualization (nl2vis) enables users to create visualizations from natural language queries, making data insights more accessible. However, nl2vis faces challenges in interpreting ambiguous queries, as users often express their visualization needs in imprecise language. To address this challenge, we introduce nvBench 2.0, a new benchmark designed to evaluate nl2vis systems in scenarios involving ambiguous queries. nvBench 2.0 includes 7,878 natural language queries and 24,076 corresponding visualizations, derived from 780 tables across 153 domains. It is built using a controlled ambiguity-injection pipeline that generates ambiguous queries through a reverse-generation workflow. By starting with unambiguous seed visualizations and selectively injecting ambiguities, the pipeline yields multiple valid interpretations for each query, with each ambiguous query traceable to its corresponding visualization through step-wise reasoning paths. We evaluate various Large Language Models (LLMs) on their ability to perform ambiguous nl2vis tasks using nvBench 2.0. We also propose Step-nl2vis, an LLM-based model trained on nvBench 2.0, which enhances performance in ambiguous scenarios through step-wise preference optimization. Our results show that Step-nl2vis outperforms all baselines, setting a new state-of-the-art for ambiguous nl2vis tasks.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://nvbench2.github.io/.

1 INTRODUCTION

Natural Language to Visualization (nl2vis) solutions democratize data exploration and analysis by enabling users to generate visualizations (vis) from natural language queries (nl) [26, 35]. While recent advances in Large Language Models (LLMs) [5, 40, 41] have significantly enhanced translation accuracy, they struggle with a fundamental challenge: *natural language ambiguity*—a single query often maps to multiple valid visualizations, each representing a different interpretation of the user's intent [2, 7, 11, 37].

In nl2vis, ambiguity is particularly complex because it arises at two levels: the **data layer**, which governs how a query selects and filters data (*e.g.,* choosing between columns or applying filters), and the **visualization layer**, which determines how the data is visually represented (*e.g.,* selecting chart types). For example, in Figure 1, the user query "Show the `gross` `trend` of comedy and action movies by year" appears straightforward but contains multiple ambiguities. At the data layer, the "`gross`" could refer to either `World_Gross` or `Local_Gross` columns in the `movies` table, while "`comedy and action`" implicitly requires filtering `Genre`. At the visualization layer, the `trend` may suggest a bar chart or a line chart, and "by `year`" implies temporal binning that is not explicitly defined.

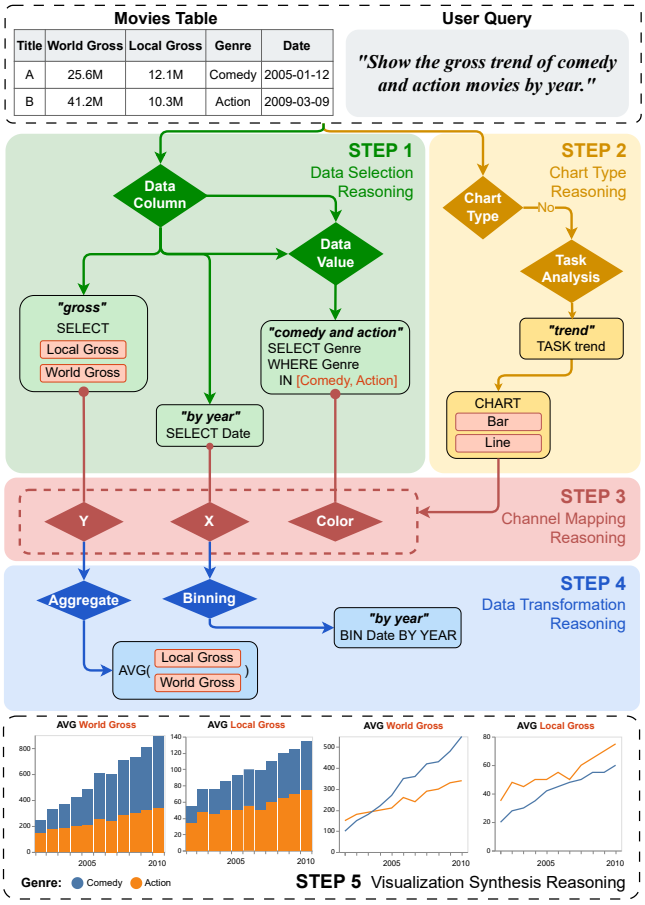

Figure 1: Example of reasoning appropriate visualizations from an ambiguous natural language query.

This example highlights how ambiguities at both the data and visualization layers interact, complicating the mapping from natural language queries to visualizations. Conventional nl2vis solutions often fail to capture these nuances, frequently yielding incomplete or overly simplistic outputs. To address this ambiguous nl2vis task, we resolve ambiguity via a human-like reasoning workflow:

Example 1 (Step-wise Disambiguation Process). *Figure 1 demonstrates our approach. Step 1 (Data Selection Reasoning) narrows "*`gross`*" to candidate columns (i.e.,* `Local_Gross` *or* `World_Gross`*)*

Table 1: Comparison of NL2VIS benchmarks.

Datasets	#-Tables	#-Samples			Chart Types	NL → VIS Mapping	NL Query Ambiguity	Reasoning Paths	NL Query Generation
		#-VIS	#-NL	#-NL/ #-VIS					
Quda [6]	36	-	14035	-	-	-	✓	✗	Human-based
NLV Corpus [32]	3	30	814	27.13	4	One/Many-to-One	✓	✗	Human-based
Dial-NVBench [29]	780	7247	124449	17.17	4	Many-to-One	✗	✗	Rule-based
VL2NL [13]	1981	1981	3962	2	7	One-to-One	✓	✗	LLM-based
VisEval [5]	748	2524	1150	0.46	4	One-to-One	✗	✗	LLM-based
nvBench [18]	780	7247	25750	3.55	4	One-to-One	✗	✗	Rule-based
nvBench 2.0	780	24076	7878	0.33	6	One-to-Many	✓	✓	LLM-based

while filtering `genres` *and* `dates` *– paralleling how analysts cross-reference schema context. Next, Step 2 (Chart Type Reasoning) evaluates task semantics ("trend") against visualization effectiveness, reflecting design best practices (using bar or line charts). Then, Step 3 (Channel Mapping) maps data fields to visual channels (e.g., X = Date, Y = Local_Gross), resolving underspecification through perceptual principles. In Step 4 (Data Transformation), it applies temporal binning and aggregation, mimicking how humans simplify temporal patterns. Finally, Step 5 (Visualization Synthesis) generates four valid outputs – a critical divergence from conventional single-output benchmarks, acknowledging real-world ambiguity tolerance.*

This multi-step process highlights the complexity of NL2VIS systems and demonstrates how a single ambiguous query can lead to multiple valid interpretations and visualizations.

Existing Benchmarks and Their Limitations. Although several benchmarks for the NL2VIS task exist [5, 6, 13, 18, 29, 32], as shown in Table 1, none explicitly evaluate how the NL2VIS systems handle ambiguity. In fact, existing efforts [5, 18, 29] often overlook this issue by adhering to the *single-correct-answer* paradigm, where each NL query maps to exactly one valid visualization. For example, nvBench [18] maps an NL query to unique visualization, ignoring more than 60% of real-world ambiguous cases [32]. Similarly, Dial-NVBench [29] supports multi-turn clarification but assumes that the final query is well-specified, which sidesteps the inherent ambiguities in natural language input.

This narrow focus leaves a critical gap in the push to advance NL2VIS systems. *How can we evaluate and improve their capacity to generate valid visualizations from ambiguous queries?*

Design Considerations. To address this challenge, a benchmark is needed that tests NL2VIS solutions on handling ambiguous queries, recognizing multiple valid interpretations, and providing appropriate visualizations. This benchmark should include diverse ambiguous queries, multiple valid outputs, reasoning paths explaining the ambiguity, and broad domain coverage.

Our Proposal. To fill this gap, we propose nvBench 2.0, the first benchmark curated for generating visualizations from ambiguous NL queries (*i.e.,* the ambiguous NL2VIS task). nvBench 2.0 includes 7,878 NL queries and 24,076 corresponding visualizations, derived from 780 tables across 153 domains. This dataset provides a robust foundation for evaluating NL2VIS solutions in scenarios where ambiguity in NL queries is a key challenge.

nvBench 2.0 meets the design considerations through a controllable ambiguity-injected NL2VIS data synthesis pipeline, which introduces ambiguity via a reverse-generation workflow. Specifically, our method starts with a seed visualization and strategically injects ambiguity (*e.g.,* variations in data selection) into its specifications. Each time ambiguity is injected, the result is a modified version of the original visualization that reflects a unique interpretation of the ambiguous query. These modified visualizations allow precise control over the types of ambiguity introduced, while ensuring that the outputs remain valid and interpretable. We then synthesize an ambiguous NL query for each set of modified visualizations, incorporating the newly injected ambiguities into a single natural language request. This process guarantees that each resulting visualization accurately represents one possible interpretation of the ambiguous query's intent.

A key benefit of our data synthesis pipeline is its transparency in handling ambiguity. By actively injecting ambiguities, we can trace how different interpretations yield distinct visualizations. For each ambiguous query, we generate reasoning paths that document the system's interpretation and the resulting valid visualization(s). This traceability enables researchers to assess both the effectiveness and interpretability of the ambiguity resolution, ensuring that the process is accurate and explainable.

Contributions. Our main contributions are summarized as follows:

- **Ambiguity-Injected NL2VIS Data Synthesizer.** We develop a data synthesizer that generates ambiguous NL2VIS data by selectively injecting ambiguities into seed visualizations, yielding multiple valid interpretations for each query while providing step-wise disambiguation reasoning paths. (Section 2)
- **nvBench 2.0 Benchmark.** We present nvBench 2.0, the first benchmark designed for the ambiguous NL2VIS task. It contains 7,878 NL queries and 24,076 corresponding visualizations, derived from 780 tables across 153 domains. Each query-visualization pair is accompanied by detailed reasoning paths, offering clear explanations of how different interpretations arise and ensuring accurate, explainable ambiguity resolution. (Section 3)
- **Step-NL2VIS for Ambiguous NL2VIS Tasks.** We propose Step-NL2VIS, an LLM-based model trained on nvBench 2.0. By leveraging step-wise preference optimization and the provided reasoning paths, Step-NL2VIS achieves the highest F1@3 (81.50%) and F1@5 (80.88%), outperforming prompting GPT-4 by 22.54% and 21.85%, respectively. (Section 4)
- **Extensive Evaluation.** We conduct comprehensive experiments to validate the effectiveness of nvBench 2.0 for training and evaluating NL2VIS systems under ambiguity. Our findings reveal the limitations of existing models when faced with ambiguous queries while demonstrating that the Step-NL2VIS outperforms baseline approaches and achieves state-of-the-art performance in ambiguous NL2VIS tasks. (Section 5)

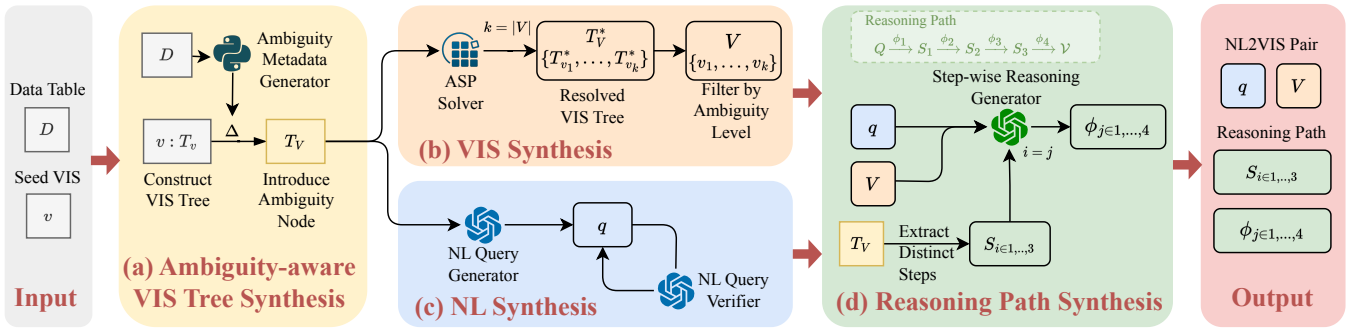

Figure 2: An overview of ambiguity-injected NL2VIS data synthesizer.

2 AMBIGUITY-INJECTED NL2VIS DATA SYNTHESIZER

2.1 Solution Overview

Figure 2 provides a high-level overview of our *Ambiguity-Injected NL2VIS Data Synthesizer*. Given a data table D and an unambiguous seed visualization v, the pipeline systematically introduces ambiguity, generates an ambiguous NL query (q) alongside a corresponding set of valid visualizations (V), and produces step-wise reasoning paths ($\mathcal{P}$) for each valid visualization. It consists of four steps:

Step 1: Ambiguity-aware VIS Tree Synthesis. We begin by constructing an initial visualization (VIS) tree T_v from D and v. Next, we introduce *ambiguity* nodes to get T_V, capturing uncertain ambiguity metadata such as data columns, mark types, and data transformations. This step ensures that each tree can branch into multiple valid interpretations. Please refer to Section 2.2 for details.

Step 2: Valid VIS Synthesis. The partially ambiguous visualization tree T_V is processed through an Answer Set Programming (ASP) solver [8], which applies visualization grammar constraints to transform the ambiguous tree into a resolved set $T_V{}^* = \{T_{v_1}{}^*, T_{v_2}{}^*, \ldots, T_{v_k}{}^*\}$, representing a set of valid visualizations $V = \{v_1, v_2, \ldots, v_k\}$. The number of resulting visualizations, $k = |V|$, indicates the ambiguity level—how many distinct interpretations the solver deems valid for the given T_V. We then select candidate visualizations based on a predefined target ambiguity level k, ensuring each retained visualization illustrates a meaningfully different way of interpreting the partially ambiguous tree. Please refer to Section 2.3 for details.

Step 3: Ambiguous NL Synthesis. We leverage an LLM-based `NL Query Generator` to synthesize an ambiguous NL query for each set of modified visualizations (*i.e.,* T_V), incorporating the newly introduced ambiguities into a single natural language request. This approach ensures that every synthesized valid visualization faithfully represents the ambiguous query's intent. Finally, an LLM-based `NL Query Verifier` checks consistency, confirming that the final NL query accurately reflects the intended ambiguities. Please refer to Section 2.4 for details.

Step 4: Ambiguity-resolved Reasoning Paths Synthesis. Finally, we produce *step-wise disambiguation reasoning paths* that document how each ambiguity is resolved in reaching every valid visualization v_i. By extracting the distinct reasoning steps from T_V to $T_V{}^*$, we offer a clear explanation of how each NL query leads to its corresponding outcome. Please refer to Section 2.5 for details.

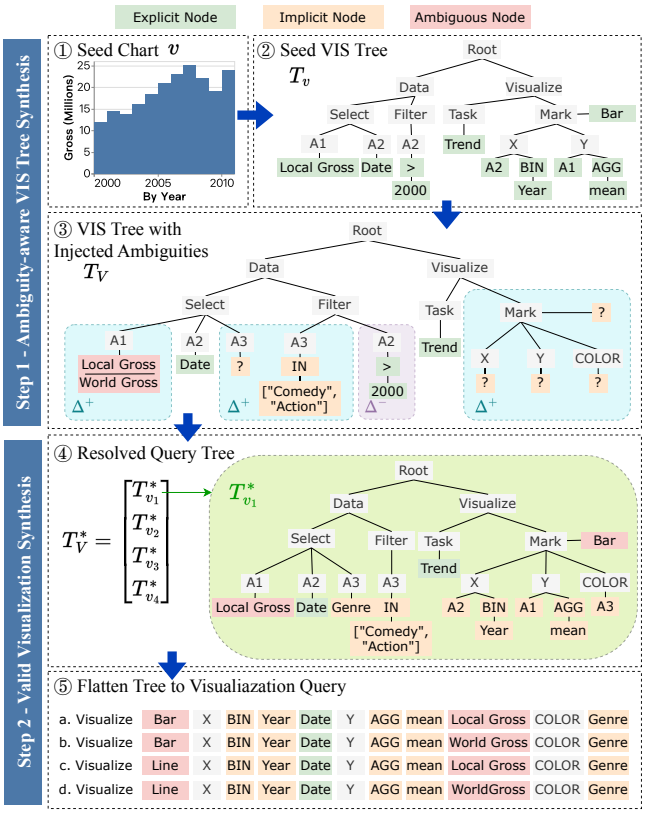

Figure 3: Injecting ambiguities into a seed visualization.

This pipeline ensures that every ambiguous aspect introduced in the VIS tree synthesis phase is clearly reflected in the final mapping of NL query to valid visualizations ($q \rightarrow V$), allowing researchers and practitioners to evaluate how effectively models handle and explain different interpretations of the same query.

2.2 Step 1: Ambiguity-aware VIS Tree Synthesis

Our ambiguity-aware visualization tree synthesis forms the foundation for synthesizing ambiguous NL2VIS data. As shown in Figure 3, this process injects ambiguities into a seed visualization.

Transforming the Seed Visualization into a Tree Abstraction. Given a data table D and a seed visualization v (*e.g.,* (Figure 3-①), we first convert the v—along with its underlying query—into an Abstract Syntax Tree (AST), which we refer to as the seed visualization tree T_v (*e.g.,* Figure 3-②). The grammar of AST is based on the predecessor work, nvBench [18]. This tree explicitly encodes all design decisions made in creating v and is formally defined as:

$$v \mapsto T_v = \{\mathbf{A} \mid \mathbf{A} = [a_1, a_2, \ldots, a_t]\} \tag{1}$$

Here, each node a_i represents a construction action for a visualization component as a tuple $(\tau, op, params)$, where:

- $\tau \in \{\texttt{explicit}, \texttt{ambiguous}, \texttt{implicit}\}$ denotes the ambiguity type of the action node;
- *op* specifies the operation (*e.g.,* data selection, chart type selection, channel mapping, data transformation selection, etc.);
- *params* contains the specific parameters for the operations.

Controlled Ambiguity Injection. We then transform T_v into an ambiguity-aware tree T_V through three operations:

- Injecting ambiguous nodes : We add nodes that represent components with multiple valid interpretations. For example, replacing "Local Gross" with an ambiguous choice between "Local Gross" and "World Gross".
- Adding implicit nodes : We include nodes for components not explicitly specified but required for visualization completion. For example, adding a node for the "COLOR" encoding channel.
- Modifying explicit nodes : We adjust certain explicit nodes to account for potential ambiguities. For example, changing a "Mark" node initially set as "Bar" into an ambiguous choice among various mark types or requiring inference from analytic tasks.

By applying these steps, the resulting ambiguity-aware tree T_V captures the full range of possible interpretations for the seed visualization. For example, as shown in Figure 3-③, this tree contains some new nodes such as:

A1 : (ambiguous, data_column, {field:[Local_Gross, World_Gross]})

A2 : (explicit, task, {value:[Trend]})

A3 : (implicit, data_value, {value:[Comedy, Action]})

Ambiguity Metadata Generation for Ambiguity Injection. To guide the injection process, we combine structured knowledge bases with LLMs to systematically identify potential semantic ambiguities in data tables. As shown in Figure 4, this metadata generation process forms the foundation for constructing ambiguity-aware visualization trees, ensuring that each node in T_V is correctly marked as ambiguous, implicit, or explicit.

Stage 1: Schema Standardization: The first step involves standardizing the original data schema by refining or expanding column names. Abbreviated or domain-specific terms are transformed into more descriptive, conventional labels. For example, a column labeled `ctry_code` is standardized to `country code`. Such standardization forms a clearer basis for subsequent ambiguity analysis.

Stage 2: Semantic Alias Discovery: After standardizing the schema, we leverage ConceptNet [31] to identify potential semantic aliases for each column name. ConceptNet's multilingual knowledge graph provides synonyms, hypernyms, and other semantically related terms, helping detect conceptual overlaps. We flag pairs of columns with similar meanings or concept overlap as potential sources of ambiguity. For example, `country code` and `city code` may both have meanings related to `area code`, introducing possible confusion in user queries.

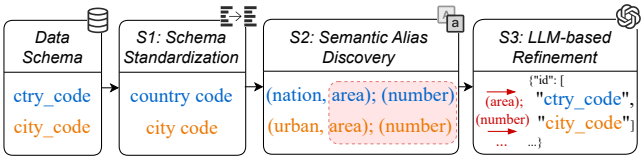

Figure 4: Ambiguity metadata generation workflow.

Stage 3: LLM-Based Refinement: We refine the flagged ambiguous column pairs using GPT-4o-mini with a chain-of-thought (CoT) prompting strategy. The model analyzes the original column names, their standardized forms (Stage 1), and the ConceptNet-derived aliases and ambiguity flags (Stage 2). It then generates a final, validated set of ambiguous pairs, which is formatted into a JSON metadata file. For example, as shown in Figure 4, one of the identified ambiguous pairs is `ctry_code` and `city_code` due to their similar word aliases. This process supports ambiguity-aware visualization generation and step-wise reasoning.

By combining these stages, we generate the necessary metadata to guide the construction of ambiguity-aware visualization trees, ensuring that each node is accurately marked as explicit, ambiguous, or implicit, thus enabling the synthesis of visualizations that reflect multiple valid interpretations of the query.

2.3 Step 2: Valid Visualization Synthesis

Once we have an ambiguity-aware visualization tree T_V, the next stage is to generate a set of valid visualizations $V = \{v_1, v_2, \ldots, v_k\}$. Each visualization v_i represents one possible resolution of the ambiguities present in T_V (see Figure 3-③). In this step, we define a resolution function $\mathcal{R}$ to systematically clarifies ambiguous and implicit nodes, transforming T_V into a set of resolved trees $\{T^*_{v_1}, T^*_{v_2}, \ldots, T^*_{v_k}\}$ (see Figure 3-④). Each resolved tree $T^*_{v_i}$ is then "flattened" into a concrete visualization query v_i (see Figure 3-⑤).

Task Description. Recap that a partially ambiguous visualization tree T_V may contain:

- *Ambiguous nodes*: Multiple valid interpretations (*e.g.,* which column to use for "gross").
- *Implicit nodes*: Necessary but unspecified details (*e.g.,* binning a date field by year).
- *Explicit nodes*: Directly specified components (*e.g.,* "bar" mark).

To produce valid visualizations, these ambiguous and implicit nodes must be resolved in a manner consistent with established visualization grammar rules (*e.g.,* requiring temporal fields to be binned). Formally, we define:

$$\mathcal{R}(T_V) \rightarrow \{T^*_{V_1}, T^*_{V_2}, \ldots, T^*_{V_k}\} \tag{2}$$

where each $T^*_{V_i}$ is a resolved tree that has no remaining ambiguity or unspecified details. The flattening process then converts each $T^*_{V_i}$ into a finalized visualization specification v_i. This yields the complete set of valid visualizations: $V = \{v_1, v_2, \ldots, v_k\}$.

In the following sections, we describe how an Answer Set Programming (**ASP**) solver [8] is used to implement the resolution function $\mathcal{R}$ while ensuring that each resolved visualization adheres to the necessary grammar constraints.

ASP Solver Objective. ASP is a declarative constraint programming paradigm well-suited for knowledge representation and reasoning [8, 20, 43]. Encoding the ambiguity resolution process and grammar rules as logical constraints has the following benefits:

- *Completeness*: The solver can enumerate all stable models (*i.e.,* all possible ways to resolve ambiguous or implicit nodes) that satisfy the visualization grammar.
- *Correctness*: Only solutions that meet mandatory constraints (*e.g.,* "temporal fields must be binned") are considered valid.
- *Diversity*: Each output corresponds to a distinct interpretation of the query, ensuring coverage of all plausible visualizations.

The number of resulting visualizations, $k = |V|$, represents the ambiguity level—how many distinct interpretations the solver deems valid for the given T_V. After obtaining these solutions, we can filter or select a subset based on a target ambiguity level k, ensuring that each retained visualization differs from the others.

ASP Syntax Overview. ASP is built on a logical foundation with several key syntactic constructs [8]. The fundamental unit in ASP is a rule of the form: `Head :- Body.`, which states that the head is true if all literals in the body are satisfied. For example, the rule: `light_on :- power_available, switch_flipped.` expresses that the light will be on if both power is available and the switch is flipped.

Some special cases include:

- *Facts*: Rules without a body represent unconditional truths. For example, `power_available.` asserts that power is available.
- *Integrity Constraints*: Rules without a head prohibit certain combinations of conditions. For example, the constraint: `:- not power_available, light_on.` ensures that the light cannot be on when power is not available.

An ASP program consists of a collection of rules, facts, and constraints that collectively define a search space. The ASP solver then computes all stable models (*i.e.,* answer sets) that satisfy these conditions. Each stable model represents a valid system state or, in our context, a valid resolution of the ambiguous visualization tree.

For example, consider a simple lighting system modeled with:

- *Rule:* `light_on :- power_available, switch_flipped.`
- *Fact:* `power_available.`, `switch_flipped.`

Given these statements, the ASP solver determines the unique answer set containing `light_on`, as all conditions in the rule body are satisfied. If we instead had `not switch_flipped.`, the solver would exclude `light_on` from the answer set.

By exhaustively computing all stable models that meet the specified constraints, the ASP solver identifies all valid visualization configurations implied by our ambiguity-aware visualization tree. This systematic resolution is key to generating a complete set of valid visualizations from an ambiguous query.

ASP Rules for Resolving Ambiguity-aware Visualization Tree. We formalize the visualization design space using ASP by converting each node in the ambiguity-aware visualization tree T_V into ASP rules. As defined in Section 2.2, each node in the visualization tree is represented as a tuple $(type, operation, parameters)$, which is mapped into ASP entities, (*e.g.,* like `entity(E, _, _).`) and their associated attributes (*e.g.,* like `attribute(A, _, _).`.

Rules for Explicit Nodes. Nodes that directly specify a visualization component are encoded as entities with fully defined attributes. For example, a node indicating a specific mark selection—such as a bar chart—is encoded in ASP as:

- `entity(mark, parent_id, mark_id).`
- `attribute((mark, type), mark_id, bar).`

These rules explicitly assert that the mark type is"bar".

Rules for Ambiguous Nodes. Nodes that allow multiple valid interpretations are encoded using ASP choice rules. For example, if an encoding node can correspond to either "temp_max" or "temp_min", we encode this ambiguity as follows:

- `1 { attribute((encoding, field), e_id, temp_max); attribute((encoding, field), e_id, temp_min) }.` ensures at least one option should be selected.
- An accompanying integrity constraint ensures that only one of the two options is selected: `:- attribute((encoding, field), e_id, temp_max), attribute((encoding, field), e_id, temp_min).`

This formulation forces the solver to choose exactly one interpretation for each ambiguous node.

Rules for Implicit Nodes. Implicit nodes represent necessary components that are not explicitly specified in the query. These nodes are encoded using placeholder attributes to indicate that the value is not determined. For example, a mark node with an unspecified chart type is represented as:

- `entity(mark, parent_id, mark_id).`
- `attribute((mark, type), mark_id, _).`

This indicates the mark exists, but its type is undetermined.

To capture the complete visualization design space, we also encode comprehensive design knowledge as ASP rules [20, 28, 43], which fall into three categories:

Definition Rules for Visualization. Declarative statements that establish foundational visualization elements, such as available chart types or encoding channels. For example, `domain((mark, type),(point; bar; pie)).` defines that the mark type for a chart can be point, bar, or pie.

Hard Constraints for Visualization. Mandatory conditions that any valid visualization must satisfy. For example, the constraint `violation(no_encodings) :- entity(mark,_,M), not entity(encoding,M,_).` ensures that every mark has at least one visual encoding channel.

Choice Rules for Visualization. Rules that govern the selection among multiple options when constructing a visualization. For example `0 { attribute((encoding, field), E, N): domain((field, name), N) } 1 :- entity(encoding,_, E).` ensures that each encoding is associated with at most one field.

Applying ASP Solver to Reason Valid Visualization. By encoding the ambiguity-aware visualization tree structure and design principles as ASP rules, we create a powerful mechanism to resolve

ambiguities. The ASP solver explores all possible resolutions for ambiguous nodes, ensuring that only solutions adhering to the visualization grammar constraints are accepted. This results in a diverse set of valid visualizations, with variations in chart type, encoding mappings, and data transformations, while staying true to the original ambiguous query.

2.4 Step 3: Ambiguous NL Query Synthesis

As shown in Figure 2 (c), this step runs in parallel with the valid visualization synthesis described in Section 2.3. Building on the ambiguity-aware visualization tree T_V, this step aims to synthesize a corresponding ambiguous natural language query q.

Task Description. Given the input ambiguous visualization tree $T_V = \{\mathbf{A} \mid \mathbf{A} = [a_1, a_2, \ldots, a_h]\}$, the corresponding natural language query q is generated using the mapping function $\mathcal{M}$:

$$Q = \mathcal{M}(T_V) = [\mathcal{M}(a_1), \mathcal{M}(a_2), \ldots, \mathcal{M}(a_h)] \tag{3}$$

where the tuple of each visualization construction action a_i in T_V is mapped to a corresponding natural language expression $\mathcal{M}(a_i)$.

For a given T_V, its corresponding q must satisfy the following conditions to ensure correctness:

- *Completeness*: Ensure that all actions in the original T_V are covered in the generated q:

$$\forall a_i \in T_V, \exists \mathcal{M}(a_i) \in Q \tag{4}$$

- *Type Preservation*: q must preserve the ambiguity types of the original action nodes:

$$\tau(\mathcal{M}(a_i)) = \tau(a_i), \quad \forall a_i \in T_V \tag{5}$$

where $\tau(a_i)$ is the ambiguity type of action node a_i.

- *Boundedness*: q should not introduce any actions outside of T_V:

$$\forall \text{ expression } e \in Q, \exists a_i \in T_V : e = \mathcal{M}(a_i) \tag{6}$$

Solution Overview. We leverage an LLM-based `NL Query Generator` to integrate the ambiguities introduced in T_V into a single and coherent query q, ensuring that the generated query faithfully reflects all the intended ambiguous components. Finally, an `NL Query Verifier` is employed to validate that q accurately captures the ambiguity without introducing any extraneous semantics. This two-step process—generation followed by verification—ensures that the final query remains consistent with the design decisions encoded in T_V while meeting the criteria of completeness, type preservation, and boundedness.

NL Diversity in Generation. NLV Corpus [32] defines several distinct categories of natural language utterances—question, command, query, and other. Since "query" somewhat overlaps with other styles, we focus on three main types: question, command, and caption, each representing a distinct style of user input:

- *Question*: Typically begins with a question word (*e.g.,* "What", "How much", "How many", etc.).
- *Command*: Usually an imperative sentence (*e.g.,* "Show a bar chart of sales by region").
- *Caption*: Includes non-standard phrases, incomplete sentences, or informal text conveying user intent, often brief (*e.g.,* "SUM (Sales) vs Date" or "budget over time").

To ensure diversity of the generated queries, we provide specific NL styles and corresponding example queries as input to the language model. These examples are randomly sampled from a large corpus to ensure variability.

NL Generator. To systematically align the structured visualization tree with diverse natural language expressions, we define explicit input-output mappings. The input to the LLM (GPT-4o-mini-turbo) delivers essential context, including data schema, sample data, action sequences, and NL style requirements. This aims to ensure that the output NL query: maintains linguistic grounding for all actions (4), preserves ambiguity types during translation (5), and avoids introducing any extraneous semantics (6). The LLM prompts are as follows:

Prompt Structure for NL Generation

(Descriptive Generation Instructions.)
Input: Data Schema D, VIS Tree T_V,
Language Style $\in \{Command, Question, Caption\}$
Output: NL Query q

NL Verifier. As indicated by recent studies [12, 39], LLMs outputs still require verification, particularly concerning *boundedness* (6). The verification can be performed by LLMs or human evaluators. In our preliminary experiments, we found that LLM-based verification is sufficient to achieve an accuracy of 99%. Thus, we designed the following prompt for LLM verification:

Prompt Structure for Verification

(Descriptive Verification Instructions.)
Input: NL Query q, VIS Tree T_v
Output Format:
- L_1: List of correct mappings $[(\text{phrase}, T_v\_node), \ldots]$.
- L_2: List of incorrect mappings $[(\text{phrase}, \text{wrong\_node}), \ldots]$.

If L_1 fully covers all nodes in T_v while L_2 remains empty, the q is considered valid and added to the dataset. Otherwise, q is classified as invalid, and it would be regenerated by the **NL Generator**. This approach checks for *completeness* (4), *type preservation* (5), and *boundedness* (6). If the verification fails, the system can regenerate the query or suggest corrections.

2.5 Step 4: Ambiguity-resolved Reasoning Path

Based on the previous discussion, we have reformulated the NL2VIS problem from a direct mapping $q \rightarrow V$ to a structured process $q \rightarrow T_V \rightarrow T_V^* \rightarrow V$. To mimic human-like reasoning workflow for ambiguity resolution, we propose decomposing the ambiguity-aware visualization generation process into a sequential reasoning path with five distinct steps, as illustrated in Figure 1:

$$q \xrightarrow{\phi_1} S_1 \xrightarrow{\phi_2} S_2 \xrightarrow{\phi_3} S_3 \xrightarrow{\phi_4} S_4 \xrightarrow{\phi_5} V \tag{7}$$

where each ϕ_i represents a reasoning function and each S_i represents the intermediate state after applying the corresponding reasoning function.

Step-①: Data Selection Reasoning. The first step parses the natural language query q into data components from the data table:

$$\phi_1(q) \rightarrow S_1 = \{a_1^c, a_2^c, \ldots, a_m^c\} \tag{8}$$

Table 2: Mappings between chart types, visual encoding channels, and analytic tasks. The encoding channels show compatible data types: C=Categorical, Q=Quantitative, T=Temporal, $\emptyset$=Not applicable.

Chart Type	Encoding Channel x|y|color|size|theta	Analytic Task
Bar	$\{C,Q,T\}\|Q\|C\|\emptyset\|\emptyset$	Trend, Distribution
Line	$\{C,Q,T\}\|Q\|C\|\emptyset\|\emptyset$	Trend, Distribution
Pie	$\emptyset\|\emptyset\|C\|\emptyset\|Q$	Distribution
Scatter	$Q\|Q\|C\|Q\|\emptyset$	Correlation
Heatmap	$\{C,Q,T\}\|\{C,Q\}\|Q\|\emptyset\|\emptyset$	Correlation
Boxplot	$\{C\}\|Q\|C\|\emptyset\|\emptyset$	Distribution

where each a_i^c represents a data component selection action, including column selection, value selection, and filter condition specification. The outcomes of this step correspond to the SELECT and FILTER nodes in the visualization tree (see Figure 3).

Step-②: Chart Type Reasoning. The second step determines appropriate visualization mark types based on the analytic task:

$$\phi_2(S_1, q) \rightarrow S_2 = S_1 \cup \{a_1^v, a_2^v, \ldots, a_n^v\} \tag{9}$$

where each a_i^v represents a visualization design action, including analytic task identification and chart type selection. As shown in Table 2, existing visualization design principles [21, 27] can establish a mapping relationship between tasks and chart types [21, 27]. When the NL query q does not explicitly specify a chart type, the identified task can guide inference—though this may introduce ambiguity as multiple chart types may be suitable for a given task. Additionally, certain tasks influence encoding channel selection in the next step.

Step-③: Channel Mapping Reasoning. The third step establishes the mappings between data components and encoding channels:

$$\phi_3(S_2) \rightarrow S_3 = S_2 \cup \{a_1^m, a_2^m, \ldots, a_p^m\} \tag{10}$$

where each a_i^m represents a channel mapping action, such as assigning data columns to encoding channels like X, Y, color, or size. This step ensures that data columns are mapped appropriately, aligning with visualization design principles, where some mapping relationships are shown in Table 2.

Step-④: Data Transformation Reasoning. The fourth step specifies necessary data transformations based on the channel mappings:

$$\phi_4(S_3) \rightarrow S_4 = S_3 \cup \{a_1^t, a_2^t, \ldots, a_r^t\} \tag{11}$$

where each a_i^t represents a data transformation action, including aggregation, binning, sorting, and filtering operations. These transformations prepare the data to be properly visualized according to the selected chart type and channel mappings.

Step-⑤: Visualization Synthesis Reasoning. The final step is to integrate all reasoning steps to generate a set of valid visualizations:

$$\phi_5(S_4) \rightarrow V = \{v_1, v_2, \ldots, v_k\} \tag{12}$$

where each v_i represents a valid visualization specification. This process can produce multiple valid visualizations that address different aspects of the ambiguity in the original query (see Figure 1).

This structured reasoning process systematically addresses ambiguity at each step while adhering to visualization design principles. Each step builds upon prior decisions, progressively refining the visualization specifications to account for multiple valid interpretations of the original NL query.

Formally, the complete reasoning path can be expressed as the composition of the step-wise reasoning functions:

$$\mathcal{F}(q, D) = (\phi_5 \circ \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1)(q, D) \rightarrow V \tag{13}$$

This decomposition simplifies the ambiguity-aware nl2vis process, breaking down complex reasoning into steps that better align with LLMs' strengths in natural language understanding and generation. Techniques like chain-of-thought prompting or step-wise direct preference optimization (step-DPO) [14, 16] can further improve LLM performance.

Finally, as shown in Figure 2 (d), the LLM-based step-wise reasoning generator takes the NL query q, the generated unambiguous visualization v, and the ambiguous visualization tree T_V as input. It then performs reverse reasoning for each step (13), generating text-based reasoning descriptions. For example, when resolving chart type ambiguity in Figure 1, the LLM reasons, "Since this query requests a trend analysis over time, either bar charts or line charts would be appropriate, as both effectively represent temporal patterns in the data" for Step-②. The LLM prompts are as follows:

Prompt Structure for Step-wise Reasoning

(Specific Reverse Reasoning Instruction for Step i.)
Input: NL Query q, VIS Set V, Vis Tree T_V,
Previous Step Answers $S1$, ..., S_{i-1}
Output: Step-wise Text-based Answer S_i

3 THE NEW BENCHMARK: NVBENCH 2.0

3.1 Synthesizing nvBench 2.0

To create our new benchmark, we apply the *Ambiguity-Injected nl2vis Data Synthesizer* (introduced in Section 2) to nvBench 1.0 [18], a large-scale, cross-domain, and unambiguous nl2vis benchmark. Table 1 summarizes the key differences between nvBench 1.0 and our newly generated dataset, nvBench 2.0.

In constructing nvBench 2.0, we retain the same data tables and seed charts as in nvBench 1.0 to ensure consistency in domain coverage and baseline complexity. However, we expand the range of chart types by additionally incorporating boxplots and heatmaps to increase the variety of possible visual encodings. This enhancement, combined with our systematic ambiguity injection, allows nvBench 2.0 to better capture the inherent fuzziness of real-world nl2vis scenarios, while maintaining the rich, multi-domain nature of the original nvBench 1.0.

3.2 Statistics of nvBench 2.0

Data Tables. Figure 5 (a.1) shows that most tables in our dataset have 2–5 columns, with fewer than 50 tables having more than 7 columns. As Figure 5 (a.2) illustrates (log scale), row counts range

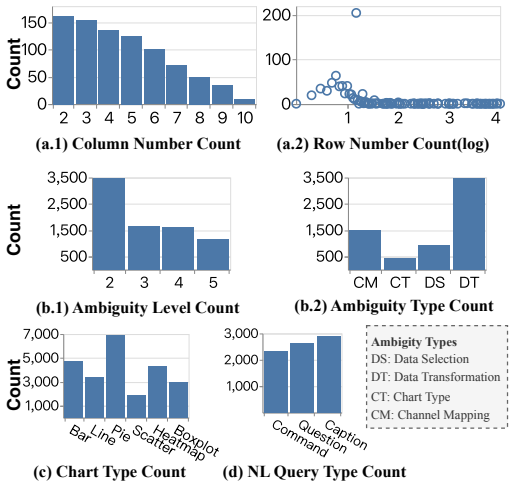

Figure 5: Statistics of nvBench 2.0.

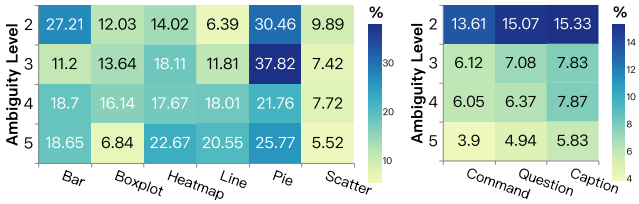

Figure 6: Left: Chart Types vs. Ambiguity Levels k. Right: NL Style vs. Ambiguity Levels k.

widely, from 20–200 rows for many tables to outliers exceeding 10,000 rows. This variety ensures that nvBench 2.0 tests system performance across both small and large datasets.

Ambiguity Types and Levels. An important contribution of nvBench 2.0 is the systematic introduction of controlled ambiguity levels. As shown in Figure 5 (b.1), the majority of samples (approximately 3,500) have an ambiguity level of 2, indicating that two valid visualizations exist. The dataset also contains a substantial number of samples with ambiguity levels of 3, 4, and 5, enabling a thorough evaluation of systems under increasingly complex ambiguous scenarios. Figure 6 further illustrates the relationship between ambiguity levels and two factors: chart types (left) and NL styles (right), showing comprehensive data coverage. Figure 5 (b.2) categorizes ambiguity by type: Data Transformation (DT) ambiguities are most prevalent (~ 3,500 examples), followed by Channel Mapping (CM) ambiguities (~ 1,500 examples), with Data Selection (DS) and Chart Type Selection (CT) ambiguities represented by approximately 900 and 400 examples, respectively.

Visualizations. Figure 5 (c.1) shows the distribution of chart types in nvBench 2.0. Pie charts are the most common, with around 6,000 examples, followed by bar charts (~4,000) and heatmaps (~3,500). Additionally, line charts (~2,800), boxplots (~2,000), and scatter plots (~1,500) are also well-represented, ensuring that the benchmark covers all major visualization types. This distribution reflects common visualization practices, where pie and bar charts are widely used for categorical comparisons, while the other types serve specialized analytical needs.

Table 3: Distribution of natural language styles across chart types and word count statistics

NL Style	Count by Chart Type						Total	Word Count		
	Bar	Line	Pie	Scatter	Boxplot	Heatmap		Avg.	Max	Min
Command	1368	922	1922	608	1319	894	2338	14.20	60	6
Question	1570	1084	2299	679	1403	966	2636	14.04	39	5
Caption	1779	1363	2651	581	1589	1079	2904	14.00	65	5
Total	4717	3369	6872	1868	4311	2939	7878	14.07	65	5

Table 4: Ambiguity count at each reasoning step

Ambiguity Type	Count	Probability
STEP 1. Data Selection (DS)	911	11.56%
STEP 2. Chart Type (CT)	428	5.43%
STEP 3. Channel Encoding (CE)	6937	88.06%
STEP 4. Data Transformation (DT)	3624	46.00%

Table 5: Statistics of ambiguity patterns.

Ambiguity Pattern	Count	Ambiguity Pattern	Count
CE	3544	CT+DT	34
DT	829	DS+DT	24
CT	41	CE+DS+DT	364
DS	13	CE+CT+DT	171
CE+DT	2190	CE+CT+DS	12
CE+DS	486	CE+CT+DS+DT	12
CE+CT	158	Total	7878

NL Queries. Figure 5 (d) presents the natural language query distribution. Command-based queries (*e.g.,* "Show me the sales by region") are most frequent (~4,000). Question-based queries (*e.g.,* "What are the sales trends?") and caption-like statements (*e.g.,* "SUM (Sales) vs Date") appear in about 2,000 and 1,800 instances, respectively. Table 3 provides a detailed breakdown of NL styles across different chart types, along with word count statistics. The average word count remains consistent (~14 words), with captions exhibiting the longest maximum length (65 words). This distribution highlights the dataset's diversity in both linguistic structure and visualization needs, ensuring that nvBench 2.0 can effectively evaluate systems' capabilities to handle diverse user interactions.

Statistics of Step-wise Reasoning Paths. A distinctive feature of nvBench 2.0 is the inclusion of structured reasoning paths that document the decision-making process for resolving ambiguities. These step-wise reasoning paths, aligned with the framework outlined in Section 2.5, provide clear explanations for the different valid interpretations associated with each ambiguous query. Table 4 presents the distribution of ambiguities across the different reasoning steps. Channel Encoding (Step 3) contains the highest concentration of ambiguities, affecting 88.06% of all samples. Data Transformation (Step 4) follows with 46.00% of samples containing ambiguities, showing that operations like aggregation and binning are also frequently underspecified in natural language queries. Data Selection (Step 1) and Chart Type (Step 2) exhibit lower ambiguity rates at 11.56% and 5.43% respectively, suggesting they are typically

more explicitly stated or easily inferred from the query context. Table 5 presents the frequency of different ambiguity patterns. The most common ambiguity type is CE (3,544 instances), followed by CE+DT (2,190), while more complex multi-category ambiguities are less frequent. Data transformation ambiguities often co-occur with other ambiguity types. This distribution underscores the prevalence of encoding-related ambiguities and the challenge of resolving overlapping ambiguity types in visualization generation tasks. These statistics highlight the complex, multi-dimensional nature of ambiguity in NL2VIS tasks and underscore the importance of step-wise reasoning for systematically resolving these ambiguities.

4 STEP-NL2VIS FOR AMBIGUOUS NL2VIS

In this section, we present Step-NL2VIS, a new model for the ambiguous NL2VIS task. Step-NL2VIS addresses ambiguity by incorporating a step-wise reasoning process, as detailed in Section 2.5, and leveraging the rich step-wise data provided by nvBench 2.0. Built on base LLMs, Step-NL2VIS is fine-tuned on nvBench 2.0 using a pipeline that aligns its outputs with the dataset's reasoning paths via supervised fine-tuning (SFT) and step-wise preference optimization (Step-DPO [14]).

4.1 Preference Optimization with Step-DPO

Previous NL2VIS methods have typically employed either prompting LLMs [36] or fine-tuning LLMs [29], where the LLM is directly tasked with generating the final VIS definition based on user-provided natural language and table schema information.

Recently, process supervision paradigms [17] and preference optimization techniques [14] have demonstrated significant advancements across various domain tasks. A pivotal aspect in validating the effectiveness of nvBench 2.0 is determining how to leverage the step-wise disambiguation reasoning paths within the nvBench 2.0 dataset to provide process supervision and enhance model performance. Consequently, we adopt the Step-DPO [14], which utilizes step-wise paired correct and incorrect samples for preference optimization, thereby delivering rich process supervision signals to the model and fostering improved accuracy at each step.

Formally, we define an input prompt x and an VIS answer y, where x includes the user natural language and table schema information, and y can be represented as $s_1 \oplus \cdots \oplus s_n$, where s_i denotes the i-th reasoning step defined in Section 2.5. Given the input x and a sequence of correct preceding reasoning steps $s_{1\sim k-1} = s_1 \oplus \cdots \oplus s_{k-1}$, Step-DPO aims to maximize the probability of the correct next reasoning step s_{win} and minimize the probability of the incorrect one s_{lose}. This objective can be formulated as:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x, s_{1\sim k-1}, s_{win}, s_{lose})\sim D_p} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(s_{win}|x, s_{1\sim k-1})}{\pi_{ref}(s_{win}|x, s_{1\sim k-1})} - \beta \log \frac{\pi_\theta(s_{lose}|x, s_{1\sim k-1})}{\pi_{ref}(s_{lose}|x, s_{1\sim k-1})} \right) \right]$$

where D_p represents a step-wise preference dataset. $\pi_\theta(\cdot|x, s_{1\sim k-1})$ denotes the policy model to be optimized, while $\pi_{ref}(\cdot|x, s_{1\sim k-1})$ refers to the reference model, which remains unchanged during the training process. The hyperparameter β controls the divergence between the optimized policy and the reference model.

4.2 Cold-start with Supervised Fine-tuning

Prior studies, such as those employing Chain-of-Thought (CoT) [38] prompting, have demonstrated the capability of LLMs to engage in step-wise reasoning through the utilization of simple "`think step-by-step`" instructions. However, under this paradigm, the planning of steps and the format of output are indiscriminate. This poses challenges in the precise extraction of answers corresponding to each individual step, and consequently, impedes the accurate alignment with the step-wise data provided within the nvBench 2.0 dataset for the purpose of validating step-level correctness. To address this limitation, we use nvBench 2.0 training set and employ Supervised Fine-Tuning (SFT) as a cold-start mechanism to facilitate the LLM's learning of our predefined step-wise output format. The specific prompt template is shown below:

Prompt Structure for Step-NL2VIS SFT

Task Description: You are a good data visualization expert. Your task is to recommend visualization charts corresponding to the ambiguous/incomplete NL Query. You need to think step by step.
Input: Table Schema, User Query
Output: <step_1><thinking>...</thinking><answer>...</answer></step_1> <step_2>...

4.3 Step-wise Preference Data Construction

A crucial aspect of Step-DPO is the acquisition of a step-wise preference dataset. As described in Section 4.1, our nvBench 2.0 dataset contains step-wise ground-truth. Therefore, we adopt an online data collection strategy. Initially, we utilize a model that has undergone Supervised Fine-Tuning (SFT) cold-start to perform inference on the nvBench 2.0 development set, yielding $D_0 = \{(x, \hat{y})\}$, where $\hat{y}$ represents the model's step-wise output, expressible as $\hat{s}_1 \oplus \cdots \oplus \hat{s}_n$. Subsequently, we conduct a step-wise evaluation comparing $\hat{y}$ with the ground-truth y, verifying the correctness of each step until the identification of the first error, and recording its corresponding step number k. We designate the erroneous step $\hat{s}_k$ as the incorrect reasoning step s_{lose}, and the ground-truth step s_k as the correct reasoning step s_{win}. The construction of the preference dataset $D_p = \{(x, \hat{s}_{1\sim k-1}, s_{win}, s_{lose}\}$ is then readily achieved through the integration of input x and previous reasoning steps $\hat{s}_{1\sim k-1}$.

5 EXPERIMENTS

In our experiments, we aim to answer two fundamental questions about ambiguous NL2VIS tasks. First, how effectively do different approaches—including state-of-the-art LLMs and our proposed Step-NL2VIS —handle visualization generation from queries with varying levels of ambiguity? Second, what impact does step-wise reasoning have on performance across different chart types and ambiguity scenarios compared to direct generation approaches?

To address these questions, we designed a comprehensive evaluation framework comparing prompting-based methods (with and without step-wise reasoning) against fine-tuning approaches. We assess performance using standard information retrieval metrics across multiple ambiguity levels (ranging from 2 to 5 interpretations) and analyze how performance varies across different visualization types and query formulations.

Table 6: Overall performance comparison between different models on nvBench 2.0.

Model	R@K(%)			P@K(%)			F1@K(%)		
	K=1	K=3	K=5	K=1	K=3	K=5	K=1	K=3	K=5
GPT-4o-mini	34.72	51.92	54.65	91.88	86.86	81.76	49.31	59.73	57.60
GPT-4o	36.56	46.35	46.79	97.07	**95.83**	**95.52**	51.96	58.96	59.03
Qwen2.5-7B	34.65	46.20	47.17	92.68	90.68	89.33	49.34	57.09	56.67
GPT-4o-mini-Step	35.13	47.68	47.91	93.48	92.54	92.08	49.96	59.29	59.10
GPT-4o-Step	36.30	48.92	49.21	96.94	95.47	95.08	51.72	60.78	60.66
Qwen2.5-7B-Step	35.20	61.86	64.08	93.61	89.26	86.23	50.05	68.56	67.76
Qwen2.5-7B-SFT	33.23	73.44	76.32	88.42	83.36	80.18	47.26	75.79	75.30
Step-nl2vis (ours)	**37.30**	**77.09**	**79.74**	**99.20**	94.27	91.17	**53.04**	**81.50**	**80.88**

5.1 Experimental Setup

Datasets. We use nvBench 2.0 as our experimental dataset, which contains a total of 7878 nl queries. We randomly split the dataset into training, development, and testing sets at a ratio of 80%, 10%, and 10%, containing 6377, 750, and 751 queries, respectively.

Methods. We conduct prompting-based methods and fine-tuning-based methods to assess the performance on ambiguous nl2vis tasks using our nvBench 2.0. The primary objective of this evaluation is to examine the model's capacity to generate diverse and semantically meaningful visualizations when confronted with ambiguous or incomplete NL queries.

Prompting-based Methods. We evaluate two prompting strategies with GPT-4o-mini, GPT-4o and Qwen2.5-7B-Instruct model:

- **Direct Prompting:** Models receive structured **Data Information** and an **NL Query**, then generate 1-5 distinct charts covering possible interpretations of ambiguous queries.
- **Step Prompting:** Models are guided to "`think step-by-step`", explicitly articulating their reasoning process before generating visualizations. Models using this approach are denoted with a "-Step" suffix in Table 6.

Supervised Fine-tuning Method.

- **Qwen2.5-7B-SFT**: We performed supervised fine-tuning on the Qwen2.5-7B-Instruct model using the training set, enabling direct generation of multiple Vega-Lite definitions without step-wise reasoning. Training involved three epochs with a global batch size of 16, a learning rate of 2e-5, the AdamW optimizer, and a cosine learning rate scheduler with 0.1 warmup ratio.

Preference Learning Method.

- **STEP-NL2VIS**: We designed Step-nl2vis to handle the ambiguity in nl2vis through step-wise reasoning as detailed in Section 4. After initial supervised fine-tuning of Qwen-2.5-7B-Instruct, we constructed a preference dataset from the nvBench 2.0 development set for Step-DPO training. This process used one epoch with a global batch size of 4, a linearly decaying learning rate from 2e-6, and the AdamW optimizer.

Evaluation Metrics. We evaluate visualization recommendations using the following standard information retrieval metrics:

- **Precision@K (P@K)**: Assesses recommendation accuracy by calculating the proportion of valid visualizations among the top-K outputs. Higher P@K indicates more trustworthy recommendations, with fewer incorrect visualizations shown to users.
- **Recall@K (R@K)**: Quantifies how completely the model covers the golden visualization space by measuring the proportion of valid visualizations successfully identified. This captures the model's ability to represent multiple valid interpretations for ambiguous queries.
- **F1@K**: Provides a balanced measure that combines precision and recall through their harmonic mean. This comprehensive metric rewards systems that achieve both high coverage of the golden answer space and high accuracy in their recommendations.

For all experiments, we report these metrics at K ∈ {1, 3, 5} to evaluate performance across different recommendation set sizes.

5.2 Experimental Results

Overall Results. Table 6 presents the comprehensive performance evaluation of different models on nvBench 2.0. Our proposed Step-nl2vis achieves state-of-the-art performance across most metrics, significantly outperforming both prompting-based and fine-tuning-based baselines. Specifically, Step-nl2vis obtains the highest F1@3 (81.50%) and F1@5 (80.88%), demonstrating its superior ability to handle ambiguity in nl2vis tasks. Step-wise reasoning consistently improves performance across all models. For both GPT-4o and Qwen2.5-7B, the "-Step" variants show notable improvements in F1 scores compared to their direct prompting counterparts. This validates our hypothesis that decomposing complex visualization reasoning into explicit steps helps resolve ambiguity more effectively. Fine-tuning on nvBench 2.0 substantially enhances recall at higher K values. Qwen2.5-7B-SFT achieves 73.44% R@3 and 76.32% R@5, significantly higher than prompt-based methods, indicating better coverage of the valid visualization space. However, it sacrifices some precision compared to prompting approaches. Our preference-optimized Step-nl2vis achieves the best balance between precision and recall. At K=1, it maintains exceptional precision (99.20%) while improving recall over all baselines, and at K=3 and K=5, it achieves substantial gains in recall without significant precision degradation.

Performance Analysis Across Chart Types. Figure 7 presents a comprehensive heatmap of F1 scores for different methods across various chart types and ambiguity levels. Several important patterns emerge from this visualization. Step-nl2vis (bottom right) consistently outperforms other models across most chart types and ambiguity levels, as indicated by the darker blue cells. These results demonstrate that the step-wise reasoning approach significantly enhances performance on ambiguous nl2vis tasks. Moreover, models incorporating step-wise reasoning (those with "-Step" suffix)

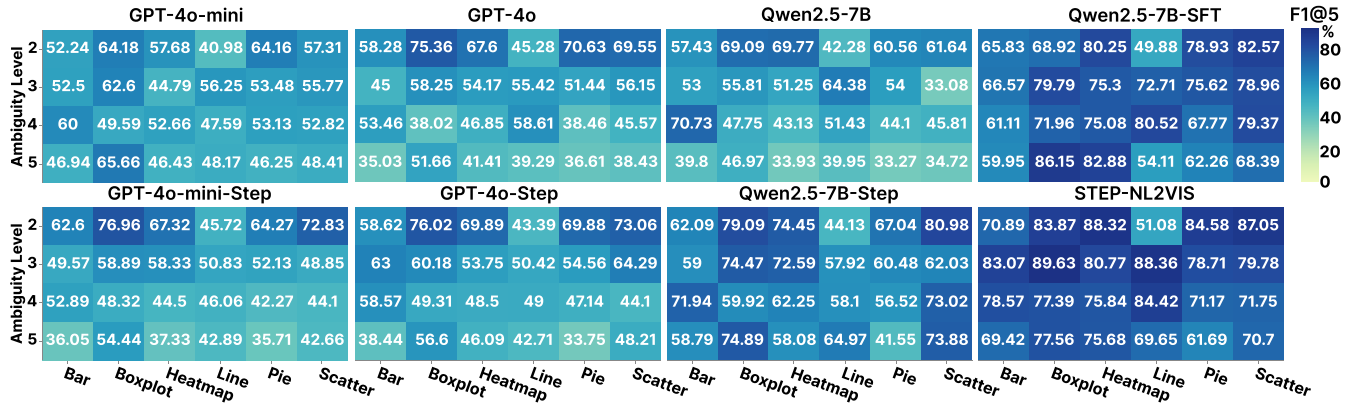

Figure 7: F1 across different models and ambiguity levels.

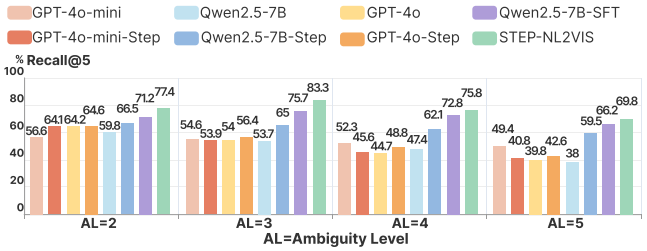

Figure 8: Recall across different models and ambiguity levels.

generally show better performance than their direct prompting counterparts, confirming the effectiveness of decomposing complex visualization reasoning into explicit steps.

- **Impact of Visualization Types.** The experimental results reveal that different chart types exhibit varying levels of difficulty for models to generate accurately. Boxplot and Scatter charts generally achieve higher F1 scores, indicating they are easier for models to handle. In contrast, Pie charts perform worse at high ambiguity levels (AL=5), while Line charts consistently show lower accuracy across all ambiguity levels, with F1 scores only around 40% to 51%, even at low ambiguity (AL=2). These findings suggest that certain visualization types pose greater challenges for model interpretation and generation.
- **Ambiguity Level Effects.** The data shows a clear degradation in performance as ambiguity level increases (from level 2 to level 5): At ambiguity level 2, most models maintain relatively high F1 scores (60-80%). By ambiguity level 5, even the best performing models struggle to maintain the same level of performance, with Qwen2.5-7B-Step achieving 41.55% and Step-nl2vis achieving 61% F1 score for pie charts at this highest ambiguity level. This pattern confirms the inherent challenge of handling highly ambiguous natural language queries.
- **Step-wise reasoning enhances performance but alters strengths for certain models.** Models such as GPT-4o-mini, GPT-4o, and Qwen2.5-7B exhibit improvements in performance when utilizing step-wise reasoning, while still maintaining their original strengths across visualization types and ambiguity levels. However, for Qwen2.5-7B, the introduction of step-wise reasoning leads to notable shifts in its area of expertise. Specifically, Qwen2.5-7B-Step demonstrates significant improvements in Boxplot (74.47%) and Heatmap (72.59%) generation at Ambiguity Level 3—an enhancement that was not prominently observed in the base Qwen2.5-7B model. This suggests that step-wise reasoning not only improves overall performance but can also reshape a model's proficiency across different visualization tasks.

Performance Analysis on Ambiguity Resolution Capacity. Figure 8 illustrates the Recall@5 metric, which quantifies each model's capability to generate valid visualizations from NL queries with varying ambiguity levels. Our proposed model, Step-nl2vis, demonstrates superior recall performance across all ambiguity levels examined. At ambiguity level 3 (AL=3), it achieves 83.3% recall, representing a statistically significant improvement over comparative approaches. Additionally, Recall@k analysis across ambiguity levels (AL=2 to AL=5) reveals some noteworthy findings:

- **Step-wise reasoning significantly enhances performance.** Models implementing step-by-step reasoning methodologies (denoted with the "-Step" suffix) consistently demonstrate superior performance compared to their non-step-wise counterparts. For instance, Qwen2.5-7B-Step exhibits markedly improved performance metrics relative to the base Qwen2.5-7B implementation.
- **Inverse correlation between performance and ambiguity.** The experimental results indicate a consistent negative correlation between recall performance and ambiguity level for the majority of models evaluated. This trend confirms the inherently increasing complexity of visualization generation as query ambiguity intensifies.
- **Maximum performance differentiation occurs at intermediate ambiguity levels.** The performance delta between evaluated models reaches its maximum at AL=3 and AL=4, suggesting these intermediate ambiguity levels provide optimal conditions for discriminating between different model capabilities.
- **Fine-tuning methods yield robust performance under increasing ambiguity.** While performance degradation is observed across all models as ambiguity increases, models employing Supervised Fine-Tuning (Qwen2.5-7B-SFT) and Preference Learning (Step-nl2vis) methodologies maintain superior performance characteristics at elevated ambiguity levels. Notably, the performance differential between Step-nl2vis and alternative approaches expands proportionally with increasing ambiguity.

Key Implications for Ambiguous NL2VIS Systems. Our experimental results reveal several important implications for the design of NL2VIS systems that can effectively handle ambiguity. First, the significant performance improvements achieved through step-wise reasoning underscore the importance of breaking down complex visualization tasks into explicit interpretable steps rather than attempting direct translation. Second, the observed performance variations across different chart types and ambiguity levels suggest that future NL2VIS systems should adaptively select reasoning strategies based on both the query characteristics and the target visualization type. Third, the superior performance of STEP-NL2VIS, particularly at higher ambiguity levels, demonstrates that preference-optimized models can learn to effectively balance precision and recall—maintaining high accuracy while capturing the full range of valid interpretations. These findings point toward a paradigm shift in STEP-NL2VIS development: from single-output systems toward multi-interpretation frameworks that explicitly model and resolve ambiguity through structured reasoning processes.

6 RELATED WORK

6.1 NL2VIS benchmarks

NL2VIS benchmarks play a crucial role in evaluating the performance of NL2VIS systems. As the predecessor of NVBENCH 2.0, nvBench [18] is a commonly used NL2VIS benchmark, constructs datasets by leveraging the semantic alignment between SQL and VQL (Visualization Query Language), which is a SQL-like specification that defines the visualization structure and details the data transformation processes. It employs template-based structures to systematically translate VQL into NL. This structured approach facilitates end-to-end model training by enhancing the clarity of both inputs and outputs [15, 19, 30, 33, 40, 42]. Building on nvBench [18], Dial-NVBench [29] introduces multi-turn dialogues, allowing models to capture user intent through iterative interactions. VisEval [4] further refines nvBench by filtering out ambiguous, irrational, duplicated, and incorrect queries using a three-step selection process (rule-based, LLM-based, and human-based), and offers an automated evaluation framework covering validity, legality, and readability. However, all three benchmarks [4, 18, 29] remain focused on well-specified queries that map directly to a single correct visualization, without explicitly addressing ambiguity in user intent.

To explore ambiguous and under-specified query formulations, ChartGPT [34] extends nvBench by prompting GPT-3 to generate more abstract and natural utterances compared to the original ones. Similarly, while some other NL2VIS datasets include ambiguous queries [6, 10, 32], they do not explicitly define ambiguity types and provide a complete set of valid chart results. Beyond the realm of NL2VIS, ambiguity has also been explored in NL2SQL benchmarks, where studies have considered data selection and computation ambiguity [2, 23], but they do not address ambiguity in the visualization space. While some NL2VIS systems have attempted to address ambiguity by detecting it [7, 24] or inferring underspecified queries [25], they lack a benchmark for systematic evaluation.

To fill this gap, we propose NVBENCH 2.0, the first ambiguity-aware NL2VIS benchmark, which provides ambiguous user queries and supports one-to-many mappings with multiple valid visualizations. By doing so, it enables a more comprehensive evaluation of NL2VIS systems in real-world scenarios.

6.2 LLMs for Data Synthesis

Recently, the use of LLMs for data synthesis or data augmentation has become increasingly prevalent. Many studies leverage LLM-generated data for training models [9, 45], as well as for evaluating the performance of other trained models [22]. In the NLP domain, researchers have utilized LLMs to generate synthetic data for tasks like text classification [1, 3, 44]). These works showcase that LLM-generated data can enhance data diversity, thereby improving model generalization and robustness. Building on this, VL2NL [13] extends LLMs to NL2VIS domain, generating NL descriptions (*e.g.,* L1 and L2 captions, and user commands) from Vega-Lite specifications.

Similarly, the application of LLMs for tabular data or database-related tasks has gained traction. Common approaches for generating NL2SQL or table question answering datasets often involve generating NL queries first, followed by SQL generation [2, 23]. ScienceBenchmark [46] takes a reverse approach by starting with seed SQL queries, then generating new queries from the domain schema, and translating them into NL using fine-tuned LLMs. We follow this reverse construction philosophy in developing NVBENCH 2.0. Specifically, we begin by extracting VQL from seed charts and then use LLMs to reverse engineer the corresponding NL descriptions. The advantage of this approach is that VQL clearly defines each step and the ambiguity types involved, allowing us to better capture one-to-many (NL, VIS) pairs.

By leveraging LLMs to generate multi-step reasoning data, the performance of models on long-chain and complex reasoning tasks can be further improved. As demonstrated by Hunter et al. [16], process supervision via multi-step reasoning significantly enhances model reliability on tasks such as mathematical problem-solving. Similarly, Step-DPO [14] shows that generating step-wise reasoning data enables models to better capture intermediate steps, resulting in improved accuracy. Following this approach, we also generate multi-step reasoning data for tasks in the NL2VIS domain, where each step of the reasoning process is explicitly defined, contributing to more accurate and interpretable model predictions.

7 CONCLUSION

In this work, we introduced NVBENCH 2.0, the first benchmark specifically designed for evaluating NL2VIS systems in scenarios involving ambiguous queries. NVBENCH 2.0 consists of 7,878 NL queries and 24,076 corresponding visualizations. These were generated through a controlled ambiguity-injection pipeline, guaranteeing valid and interpretable results while offering transparent reasoning pathways. By using NVBENCH 2.0, we offer a robust framework to assess NL2VIS systems' ability to handle ambiguities that arise in real-world applications.

We also proposed STEP-NL2VIS, an LLM-based NL2VIS model trained on NVBENCH 2.0, which significantly improves NL2VIS performance in ambiguous scenarios by applying step-wise preference optimization. Our experimental results demonstrate that STEP-NL2VIS outperforms all existing baselines, establishing a new state-of-the-art for handling ambiguity in NL2VIS tasks.

REFERENCES

[1] Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2019. Not Enough Data? Deep Learning to the Rescue! *arXiv e-prints*, Article arXiv:1911.03118 (Nov. 2019), arXiv:1911.03118 pages. https://doi.org/10.48550/arXiv.1911.03118 arXiv:1911.03118 [cs.CL]

[2] Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. 2023. Benchmarking and Improving Text-to-SQL Generation under Ambiguity. In *EMNLP*. Association for Computational Linguistics, 7053–7074.

[3] Maximillian Chen, Alexandros Papangelis, Chenyang Tao, Andrew Rosenbaum, Seokhwan Kim, Yang Liu, Zhou Yu, and Dilek Z. Hakkani-Tür. 2022. Weakly Supervised Data Augmentation Through Prompting for Dialogue Understanding. *ArXiv* abs/2210.14169 (2022). https://api.semanticscholar.org/CorpusID:253107809

[4] Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. 2024. VisEval: A Benchmark for Data Visualization in the Era of Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 31 (2024), 1301–1311. https://api.semanticscholar.org/CorpusID:270869570

[5] Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. 2025. VisEval: A Benchmark for Data Visualization in the Era of Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 31, 1 (2025), 1301–1311. https://doi.org/10.1109/TVCG.2024.3456320

[6] Siwei Fu, Kai Xiong, Xiaodong Ge, Siliang Tang, Wei Chen, and Yingcai Wu. 2020. Quda: Natural Language Queries for Visual Data Analytics. *arXiv e-prints*, Article arXiv:2005.03257 (May 2020), arXiv:2005.03257 pages. https://doi.org/10.48550/arXiv.2005.03257 arXiv:2005.03257 [cs.CL]

[7] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 489–500. https://doi.org/10.1145/2807442.2807478

[8] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming* 19, 1 (2019), 27–82.

[9] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 3309–3326. https://doi.org/10.18653/v1/2022.acl-long.234

[10] Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, Zejian Yuan, and Dongmei Zhang. 2023. Text2Analysis: A Benchmark of Table Question Answering with Advanced Data Analysis and Unclear Queries. arXiv:2312.13671 [cs.CL] https://arxiv.org/abs/2312.13671

[11] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2018. Applying Pragmatics Principles for Interaction with Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 309–318. https://doi.org/10.1109/TVCG.2017.2744684

[12] Che Jiang, Biqing Qi, Xiangyu Hong, Dayuan Fu, Yang Cheng, Fandong Meng, Mo Yu, Bowen Zhou, and Jie Zhou. 2024. On Large Language Models' Hallucination with Regard to Known Facts. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). ACL, Mexico City, Mexico, 1041–1053. https://doi.org/10.18653/v1/2024.naacl-long.60

[13] Hyung-Kwon Ko, Hyeon Jeon, Gwanmo Park, Dae Hyun Kim, Nam Wook Kim, Juho Kim, and Jinwook Seo. 2024. Natural Language Dataset Generation Framework for Visualizations Powered by Large Language Models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 843, 22 pages. https://doi.org/10.1145/3613904.3642943

[14] Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-DPO: Step-wise Preference Optimization for Long-chain Reasoning of LLMs. *arXiv e-prints*, Article arXiv:2406.18629 (June 2024), arXiv:2406.18629 pages. https://doi.org/10.48550/arXiv.2406.18629 arXiv:2406.18629 [cs.LG]

[15] Shuaimin Li, Xuanang Chen, Yuanfeng Song, Yunze Song, and Chen Zhang. 2024. *Prompt4Vis: Prompting Large Language Models with Example Mining and Schema Filtering for Tabular Data Visualization.* Vol. 1. Association for Computing Machinery. arXiv:2402.07909 http://arxiv.org/abs/2402.07909

[16] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's Verify Step by Step. *ArXiv* abs/2305.20050 (2023). https://api.semanticscholar.org/CorpusID:258987659

[17] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. https://openreview.net/forum?id=v8L0pN6EOi

[18] 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 *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 1235–1247. https://doi.org/10.1145/3448016.3457261

[19] Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2022. Natural Language to Visualization by Neural Machine Translation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (jan 2022), 217–226. https://doi.org/10.1109/TVCG.2021.3114848

[20] Dominik Moritz, Chenglong Wang, Greg L Nelson, Halden Lin, Adam M Smith, Bill Howe, and Jeffrey Heer. 2018. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 438–448.

[21] Arpit Narechania, Arjun Srinivasan, and John Stasko. 2021. NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 369–379. https://doi.org/10.1109/TVCG.2020.3030378

[22] Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive Testing and Debugging of NLP Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 3253–3267. https://doi.org/10.18653/v1/2022.acl-long.230

[23] Irina Saparina and Mirella Lapata. 2024. AMBROSIA: A Benchmark for Parsing Ambiguous Questions into Database Queries. *ArXiv* abs/2406.19073 (2024). https://api.semanticscholar.org/CorpusID:270764916

[24] Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A Natural Language Interface for Visual Analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 365–377. https://doi.org/10.1145/2984511.2984588

[25] Vidya Setlur, Melanie Tory, and Alex Djalali. 2019. Inferencing underspecified natural language utterances in visual analysis. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 40–51. https://doi.org/10.1145/3301275.3302270

[26] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2023. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2023), 3121–3144. https://doi.org/10.1109/TVCG.2022.3148007

[27] Leixian Shen, Enya Shen, Zhiwei Tai, Yiran Song, and Jianmin Wang. 2021. TaskVis: Task-oriented Visualization Recommendation. In *Proceedings of the 23th Eurographics Conference on Visualization, EuroVis'21*. Eurographics, 91–95. https://doi.org/10.2312/evs.20211061

[28] Leixian Shen, Enya Shen, Zhiwei Tai, Yihao Xu, Jiaxiang Dong, and Jianmin Wang. 2022. Visual Data Analysis with Task-Based Recommendations. *Data Science and Engineering* 7, 4 (2022), 354–369. https://doi.org/10.1007/s41019-022-00195-3

[29] Yuanfeng Song, Xuefang Zhao, and Raymond Chi-Wing Wong. 2023. Marrying Dialogue Systems with Data Visualization: Interactive Data Visualization Generation from Natural Language Conversations. arXiv:2307.16013 [cs.AI] https://arxiv.org/abs/2307.16013

[30] Yuanfeng Song, Xuefang Zhao, Raymond Chi-wing Wong, and Di Jiang. 2022. RGVisNet: A Hybrid Retrieval-Generation Neural Framework Towards Automatic Data Visualization Generation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 1646–1655. https://doi.org/10.1145/3534678.3539330

[31] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: an open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, California, USA) *(AAAI'17)*. AAAI Press, 4444–4451.

[32] Arjun Srinivasan, Nikhila Nyapathy, Bongshin Lee, Steven M. Drucker, and John Stasko. 2021. Collecting and Characterizing Natural Language Utterances for Specifying Data Visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 464, 10 pages. https://doi.org/10.1145/3411764.3445400

[33] Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. 2024. ChartGPT: Leveraging LLMs to Generate Charts from Abstract Natural Language. *IEEE Transactions on Visualization and Computer Graphics* 14, 8 (2024), 1–15. https://doi.org/10.1109/TVCG.2024.3368621 arXiv:2311.01920

[34] Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. 2025. ChartGPT: Leveraging LLMs to Generate Charts From Abstract Natural Language. *IEEE Transactions on Visualization and Computer*

Graphics 31, 3 (2025), 1731–1745. https://doi.org/10.1109/TVCG.2024.3368621

[35] Henrik Voigt, Ozge Alacam, Monique Meuschke, Kai Lawonn, and Sina Zarrieß. 2022. The Why and The How: A Survey on Natural Language Interaction in Visualization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, Stroudsburg, PA, USA, 348–374. https://doi.org/10.18653/v1/2022.naacl-main.27

[36] Lei Wang, Songheng Zhang, Yun Wang, Ee-Peng Lim, and Yong Wang. 2023. LLM4Vis: Explainable Visualization Recommendation using ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*. ACL, Stroudsburg, PA, USA, 675–692. https://doi.org/10.18653/v1/2023.emnlp-industry.64

[37] 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 Transactions on Visualization and Computer Graphics* 29, 1 (2023), 1222 – 1232. https://doi.org/10.1109/TVCG.2022.3209357

[38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.). http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[39] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large Language Models are Better Reasoners with Self-Verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). ACL, Singapore, 2550–2575. https://doi.org/10.18653/v1/2023.findings-emnlp.167

[40] Yang Wu, Yao Wan, Hongyu Zhang, Yulei Sui, Wucai Wei, Wei Zhao, Guandong Xu, and Hai Jin. 2024. Automated Data Visualization from Natural Language via Large Language Models: An Exploratory Study. In *Proceedings of the ACM on Management of Data*. ACM, 1–28. https://doi.org/10.1145/3654992

[41] Yifan Wu, Lutao Yan, Leixian Shen, Yunhai Wang, Nan Tang, and Yuyu Luo. 2024. ChartInsights: Evaluating Multimodal Large Language Models for Low-Level Chart Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. ACL, Stroudsburg, PA, USA, 12174–12200. https://doi.org/10.18653/v1/2024.findings-emnlp.710

[42] Zhengkai Wu, Vu Le, Ashish Tiwari, Sumit Gulwani, Arjun Radhakrishna, Ivan Radiček, Gustavo Soares, Xinyu Wang, Zhenwen Li, and Tao Xie. 2022. NL2Viz: natural language to visualization via constrained syntax-guided synthesis. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, New York, NY, USA, 972–983. https://doi.org/10.1145/3540250.3549140

[43] Junran Yang, Péter Ferenc Gyarmati, Zehua Zeng, and Dominik Moritz. 2023. Draco 2: An extensible platform to model visualization design. In *2023 IEEE Visualization and Visual Analytics (VIS)*. IEEE, 166–170.

[44] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021. GPT3Mix: Leveraging Large-scale Language Models for Text Augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Punta Cana, Dominican Republic, 2225–2239. https://doi.org/10.18653/v1/2021.findings-emnlp.192

[45] Ann Yuan, Daphne Ippolito, Vitaly Nikolaev, Chris Callison-Burch, Andy Coenen, and Sebastian Gehrmann. 2021. SynthBio: A Case Study in Human-AI Collaborative Curation of Text Datasets. *arXiv e-prints*, Article arXiv:2111.06467 (Nov. 2021), arXiv:2111.06467 pages. https://doi.org/10.48550/arXiv.2111.06467 arXiv:2111.06467 [cs.CL]

[46] Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. 2023. ScienceBenchmark: A Complex Real-World Benchmark for Evaluating Natural Language to SQL Systems. *Proc. VLDB Endow.* 17, 4 (Dec. 2023), 685–698. https://doi.org/10.14778/3636218.3636225