Inferring Visualization Intent from Conversation

Haotian Li*
HKUST
Hong Kong SAR, China
haotian.li@connect.ust.hk

Nithin Chalapathi*
UC Berkeley
Berkeley, CA, USA
nithinc@berkeley.edu

Huamin Qu
HKUST
Hong Kong SAR, China
huamin@cse.ust.hk

Alvin Cheung
UC Berkeley
Berkeley, CA, USA
akcheung@berkeley.edu

Aditya G. Parameswaran
UC Berkeley
Berkeley, CA, USA
adityagp@berkeley.edu

Abstract

During visual data analysis, users often explore visualizations one at a time, with each visualization leading to new directions of exploration. We consider a conversational approach to visualization, where users specify their needs at each step in natural language, with a visualization being returned in turn. Prior work has shown that visualization generation can be boiled down to the identification of visualization intent and visual encodings. Recognizing that the latter is a well-studied problem with standard solutions, we focus on the former, i.e., identifying visualization intent during conversation. We develop Luna, a framework that comprises a novel combination of language models adapted from BERT and rule-based inference, that together predict various aspects of visualization intent. We compare Luna with other conversational NL-to-visualization and NL-to-SQL approaches (adapted to visualization intent), including GPT-3.5 and GPT-4, and demonstrate that Luna has 14.3% higher accuracy than the state-of-the-art. We also apply Luna to a usage scenario on a dataset of police misconduct, showcasing its benefits relative to other approaches.

CCS Concepts

• **Human-centered computing** → **Visualization**; • **Information systems** → **Data management systems**.

Keywords

Conversational Natural Language Interface, Visualization Recommendation

ACM Reference Format:
Haotian Li, Nithin Chalapathi, Huamin Qu, Alvin Cheung, and Aditya G. Parameswaran. 2024. Inferring Visualization Intent from Conversation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3627673.3679589

*Both authors contributed equally to this paper.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679589

1 Introduction

Visual data analysis, while essential for making sense of data, is still challenging for the vast majority of users. While programmatic visualization tools such as Vega-Lite [32] or Matplotlib [15], provide great flexibility in selecting what and how to visualize, they have a high learning curve. Meanwhile, other visual business intelligence tools such as Tableau [5] and PowerBI [3], require users to navigate complex user interfaces to build visualizations. "No-code" interfaces such as natural language have thus emerged as an alternative that allow users to build visualizations without programming experience. Such *NL2Vis* systems have been proposed to infer user intent based on a natural language question, and using that to suggest appropriate visualizations, e.g., Chat2Vis [25], and ncNet [22].

However, prior work ignores the fact that visual data analysis is *iterative*, where users build on findings from previous steps [16, 19, 41]. We therefore consider a *conversational* approach to visualization. Our scenario of a data journalist studying police misconduct in Fig. 1 Cols. 1 and 2 shows that a conversational feedback loop between system generated visualizations and user requests is well-suited for visual data analysis To address this need, there have been some systems for generating visualizations during conversation, employing rule-based approaches [12, 26, 28]. However, their rule-based nature limits flexibility and makes them brittle. For example, NL4DV [28], the newest such system cannot handle cases where the column name is not described exactly as in the input schema, as shown in Figs. 1(a) and (e), Col. 4.

In this paper, we propose a robust approach for conversational generation of visualization. Prior work [20, 42, 47] has shown that generating visualizations can be divided into two distinct steps: identifying data-specific visualization intent (i.e., *visualized attributes* and *data filters*) and determining visual encodings (i.e., how the data should be visualized). The latter can be inferred based on best practices [23, 24, 27] given the former. We therefore focus on **inferring visualization intent from conversation**. One approach is to leverage natural language to SQL (*NL2SQL*) models that translate user intent from natural language into SQL queries using a single "end-to-end" deep learning model such as CD-Seq2Seq [40, 46], R^2SQL [14], and PICARD [33]. However, these approaches cannot guarantee that syntactically and semantically correct SQL queries are produced, making extracting visualization intent from their generated SQL queries challenging. Indeed, such models fail in four of the six steps in Fig. 1, Col. 5. Yet another approach is to use general-purpose large language models (LLMs), such as GPT-3.5 [2]

and GPT-4 [30]. While these LLMs seem to perform better than other baseline approaches in Fig. 1, they still suffer from the issue of losing context in conversations and inferring incorrect intent.

Based on the insights from prior NL2Vis and NL2SQL tools, we identify multiple technical challenges in inferring visualization intent and propose a new framework called Luna to address the challenges effectively. First, the output intent should always be system-interpretable to guarantee that visualizations can be shown to users at each step. To tackle the challenge, we apply a *divide-and-conquer* strategy, instead of end-to-end as in PICARD and GPT models, for predicting users' intent. We break down the task of predicting intent (i.e., visualized attributes and data filters) into multiple sub-tasks (see Fig. 2), each learned via a specialized module. To predict the attributes to be visualized, we first determine the number of attributes and then use this to select the attributes that best associate with the users' query. Similarly, the filter attributes are inferred using two steps: determining the number of filters and then the attribute in each filter. Finally, the operators and values in filters are predicted accordingly for each filter attribute. This breakdown makes learning straightforward and guarantees syntactically correct and always-executable intents. Furthermore, we can select different neural architectures or inference algorithms for each sub-task. Secondly, the tool should understand context in conversations (i.e., users' previous questions and systems' responses) to facilitate the iterative nature of data analysis and the ambiguity in natural languages. We handle these issues using a carefully designed combination of language models and rule-based methods for each sub-task, and a specifically designed input format to summarize the context in conversations.

We evaluate Luna by comparing it against NL4DV [26], two state-of-the-art NL2SQL models (two PICARD variants with T5-3B and T5-Large [31] and CD-Seq2Seq), and two state-of-the-art LLMs (GPT-3.5 [2] and GPT-4 [30]). Our approach achieves 57.31% accuracy on the test set—a 14.3% improvement over the state-of-the-art, PICARD with T5-Large and 27.72% over GPT-4, the best performing general-purpose LLM. Furthermore, Luna has a lower GPU memory consumption and inference time than PICARD, the state-of-the-art baseline approach. Overall, our contributions are:

- We divide visualization intent inference into six sub-tasks to ensure the generation of always valid visualization intent, produce simpler sub-tasks, and enable flexible choices for the structure of the module addressing each sub-task (Sec. 3).
- Following the task breakdown, we design Luna, which leverages multiple specialized modules for inferring different aspects of visualization intent (Sec. 4).
- We evaluate Luna through a quantitative comparison with the state-of-the-art, evaluation of individual modules, and a real-world application using a police misconduct dataset. Luna outperforms other approaches with a 14.3% improvement in accuracy with only 0.1% of the inference time (Secs. 5 and 6).

2 Motivating Example

Suppose our data journalist Ada wants to study incidents of police misconduct using a dataset from the California Reporting Project (CRP) [11]. This use case is based on our collaboration as part of the CLEAN (Community Law Enforcement Accountability Network) consortium, with journalists and public defenders, to investigate police misconduct through data. One of its relations (many columns/rows omitted) is in Table 1 that records the time/location of each incident, and if weapons were found. As a programming novice, Ada uses a conversational interface, powered by Luna, as shown in Fig. 1. Luna identifies the data-specific visualization intent comprising of the visualized attributes and filters to be applied.

Table 1: Examples of incidents from the CRP dataset

id	weapon_found	city	county	day	month	year
240	yes	Bakersfield	Kern	25	6	2016
465	no	Richmond	Contra Costa	9	9	2019
841	no	Oakland	Alameda	17	3	2014

To start her exploration, Ada first inspects the locations of the incidents by asking *"Where do the cases happen?"* and receives a chart shown in Fig. 1(a) Col. 2. Based on the question, Luna infers that Ada wants to plot the distribution of cases in different cities (i.e., the visualized attribute intent is city, without any filters). She then wants to see the distribution over counties and asks *"In which counties do the cases happen?"*. Ada notices Contra Costa County has the most cases from the chart (Fig. 1(b)), wants to monitor the temporal changes of such incidents, and asks *"How do the cases change over the years?"*. Seeing that 2016 was a turning point (Fig. 1(c)), she then asks *"Can you show me the distribution of cases where weapons are found or not?"*. The generated chart (Fig. 1(d))—where the intent involves just the `weapon_found` attribute—reveals that most cases do not involve weapons. Ada wonders if this the same trend continues after 2016, asks *"What about those cases happened after 2016?"*, and receives Fig. 1(e)—with the intent now including a filter on year. She notices a minor improvement after 2016, and drills down to Contra Costa County again by asking *"Then can you also show the numbers of those cases in Contra Costa County?"*. The results are shown in Fig. 1(f), where an additional filter on county is added.

For comparison, Fig. 1 also shows other approaches: NL4DV [28], a rule-based conversational NL2Vis tool, PICARD [33], a state-of-the-art conversational NL2SQL approach built on T5 [31], Chat2Vis [25], a conversational interface built on GPT-3.5 [2], and general-purpose LLMs (GPT-3.5 [2] and GPT-4 [30]). We provide the visualization intent and corresponding visualizations for each approach. Since Chat2Vis does not expose the code for generating visualizations, we only provide the returned visualizations.

As shown in Fig. 1, end-to-end generative models such as PICARD and Chat2Vis **often fail to generate syntactically valid code.** They generate non-executable code (Fig. 1(c)-(f)), nonsensical visualizations (Figs. 1(b)), or erroneous SQL (Figs. 1(c)-(d) where non-aggregated columns are returned and Fig. 1 (f) where single instead of double quotations is used for string literals—which is not permitted according to the SQL-92 standard. PICARD also generates imaginary filter values, such as "Yes" in Fig. 1(f), where the values in the column are "yes" and "no." The same happens to the LLMs where both GPT models return "Contra Costa County" as the filter value in Fig. 1(f), leading to an empty result. Second, **rule-based approaches**, such as NL4DV, **are ineffective in dealing with the ambiguity of natural language**. Figs. 1(a) and (e) show examples where no attribute is explicitly mentioned. NL4DV fails on this case as it requires an exact match of attribute names. The failure in Fig. 1(e) further leads to a missing filter on year in Fig. 1(f).

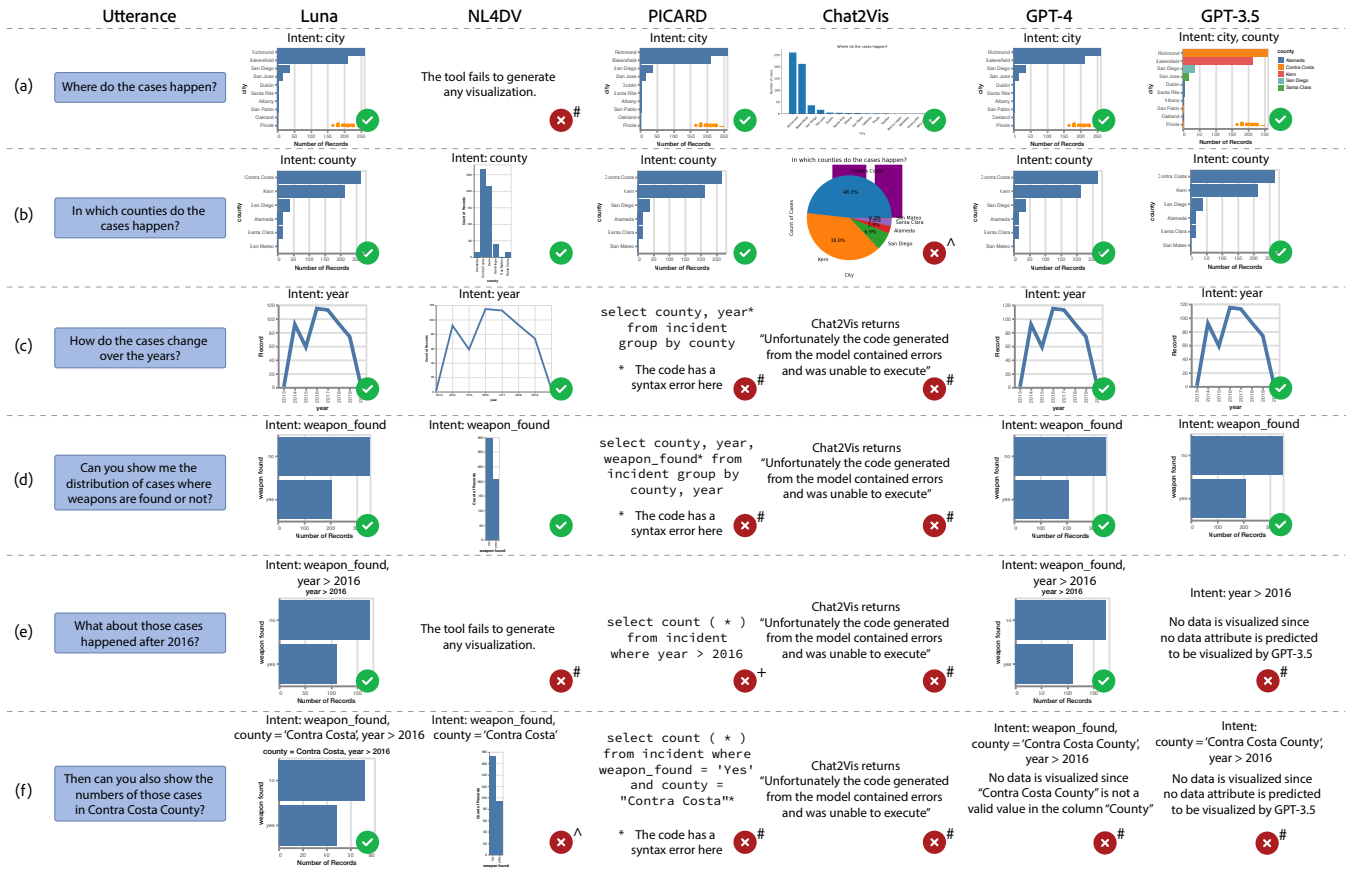

Figure 1: Illustration of a data journalist's input questions and results from different approaches. A green tick means the result matches their intent. A red cross means the result is problematic: # means the result is not executable and no visualization is generated. ∧ indicates that the visualization is incorrect. + means that a number is returned, which is unsuitable for visualization.

Finally, **existing approaches fail to handle conversational aspects.** In Fig. 1(e), PICARD fails to understand the relationship between the new and previous questions as it forgets the attribute `weapon_found` in the new SQL query even though the words "what about" and "those" refers to the previous conversation. GPT-3.5 also suffers from a similar issue. Furthermore, PICARD also misbehaves in Fig. 1(f) by adding an unnecessary filter `weapon_found = 'Yes'` and removing `year > 2016`. This real-world scenario shows the challenges in applying existing approaches to conversational visual data exploration. Our quantitative evaluation in Sec. 6 provides a comprehensive comparison among the approaches.

Thus, an effective approach to infer users' visualization intent from conversation should: (1) *produce executable results*, (2) *handle ambiguity in natural language*, and (3) *deal with conversations*.

3 Problem Formulation

Our goal is to generate visualizations from a *dataset* consisting of *records*, where each record has multiple *attributes*. At each step or *interaction* of the *conversation*, the user issues a natural language *question* or *utterance* to the conversational interface and receives a visualization in return: each question or response is called a *turn*.

We break down the generation of a visualization into two steps as in prior work [20, 42, 47]: identification of *visualization intent*, and of *visualization encodings*. The former comprises all data-centric aspects, including the attributes to be visualized and the filters should be applied. Each filter comprises an attribute, an operator, and a value. For example, in Fig. 1(f), in the intent found by Luna, the attribute to be visualized is `weapon_found`, while the filters are `county = 'Contra Costa'` and `year > 2016`. One can map the filters to the `WHERE` clause of a SQL query, while the attributes map to the `SELECT` clause. There has been extensive prior work on determining the visualization encoding given the intent, e.g., [23, 24, 27]. For example, Fig. 1(f) shows a bar chart as `weapon_found` is a Boolean field. We thus focus on determining visualization intent.

Luna addresses the challenges discussed in Sec. 2 by breaking down visualization intent prediction into sub-tasks and leveraging different modules for each, rather than a single, often brittle, model (see Sec. 4). In particular, the attributes and filter values are drawn from the dataset. This way, the predicted visualization intent is always executable. Next, we describe how Luna leverages LLMs as the backbone for each module, with carefully designed output heads. We leverage LLMs' ability in understanding natural language, thereby eliminating brittle heuristics and handling

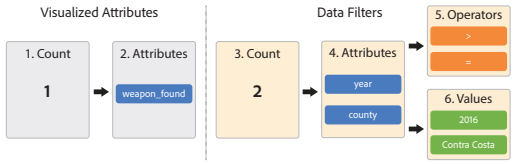

Figure 2: The six sub-tasks in predicting users' intent.

language ambiguity. Finally, we further design the input format which succinctly encodes the previous intent to enable the model to understand the context in conversations, instead of concatenating all previous interactions in prior NL2SQL systems.

4 Design of Luna

We next describe how we designed Luna to address the challenges in inferring visualization intent from conversation.

4.1 Overview

We divide the visualization intent inference problem into six sub-tasks (Fig. 2) as inspired by prior work [20]. The first two tasks predict the number of attributes and filters. The next two tasks leverage these numbers to select attributes for visualization and in the filters In the final two tasks, the other elements in filters (i.e., operators and values) are predicted based on the attributes.

Compared to end-to-end models (e.g., [33]), Luna's design provides three benefits. First, **the outputs generated by Luna will always produce a valid visualization intent**, while for prior work based on auto-regressive models, only around 36% SQL queries generated by PICARD [33] with T5-3B [31] are syntactically or semantically valid as our experiments show (see Sec. 6.1.1). Second, **our breakdown simplifies the task for each module**, as it only needs to predict a subset of the final output, e.g., data attributes, filter values or operators. This makes the models easier to train and achieve higher accuracy than a single end-to-end model, as Sec. 6 shows. Lastly, **our approach provides the flexibility to select different model architectures for different sub-tasks**. For example, we utilize a classification head rather than an attention-based head to predict the number of visualized attributes and then select the top-related attributes ranked by an attention-based head. Our experimental results verify the effectiveness of utilizing different model structures for different sub-tasks.

The architecture of Luna is shown in Fig. 3, comprising of five fine-tuned BERT-based modules followed by a heuristic-based module for filter value selection (Fig. 3). To start, the *visualized attribute count prediction* module first predicts the number of data attributes (k_a) to be visualized. Then, the concrete data attributes are selected according to the ranking of their relevance to users' interactions, which is computed by the *visualized attribute ranking* module. Similarly, a separate *filter count prediction* module determines the number of data filters (k_f) pertinent to the user utterance. Then, the attributes of the filters are selected as the top-k_f relevant attributes computed by the *filter attribute ranking* module. Each data column is then fed into the *filter operator prediction* module which selects one of four operators: $<$, $>$, $=$, and $\neq$. Finally, the *filter value prediction* module identifies the values in filters by matching words with the utterances with the distinct values in the column.

4.2 Basic Structure of LLM-powered Modules

Luna consists of five LLM-powered modules where the LLMs are applied as backbones to understand the conversations and semantic meanings of data columns and generate embeddings for prediction.

Specifically, we fine-tuned a pre-trained standard BERT-base model with 110 million parameters. While BERT is an older and lightweight model by today's standards, we find that Luna with BERT outperforms these state-of-the-art, much larger models (e.g., GPT models) in prediction accuracy (see Sec. 6). The input to the LLMs in the different modules follows the same format, derived from BERT's default. The input is formatted as the previous intent (if any exists), followed by the new utterance and the columns of the dataset. Specifically, we summarize *all* previous intents in the conversation into a single summary consisting of previous attributes and filters. We use *"previous attributes: "* and *"previous filters: "* to indicate that the two parts reveal different aspects of intent. Furthermore, operators are transformed into natural language as "is," "is not," "is larger than," etc.

To illustrate, consider part of the conversation shown in Sec. 2, where each utterance is labeled with its number in bold.

(1) Can you show me the distribution of cases where weapons are found or not? **(1)**
(2) What about those cases happened after 2016? **(2)**
(3) Then can you also show the numbers of those cases in Contra Costa County? **(3)**

The input of the last utterance to Luna has two parts:

(1) Previous intent and the new utterance: *previous attributes: weapon found. [SEP] previous predicates: year is larger than 2016. [SEP] Then can you also show the numbers of those cases in Contra Costa County?*
(2) Columns: *id [SEP] weapon found [SEP] city [SEP] county [SEP] day [SEP] month [SEP] year*

In contrast, if we instead concatenate all utterances together, like PICARD, the first parts of the input will be: *"Can you show me the distribution of cases where weapons are found or not? [SEP] What about those cases happened after 2016? [SEP] Then can you also show the numbers of those cases in Contra Costa County?"* To understand the context of the last query, the model needs to correctly understand all three utterances and identify that the word *"cases"* in the next two utterances refers to *"cases where weapons are found or not"* in the first utterance. As Figs. 1 (e)-(f) shows, PICARD fails to correctly interpret such references when generating new SQL queries, showing that it is non-trivial for models to understand the relationship between multiple utterances. This observation shows the need for a new input format in Luna to facilitate context understanding.

Furthermore, our approach is beneficial in settings involving a human in the loop. In such settings, it is challenging to ensure that the system can understand every utterance correctly. Therefore, users may need to correct the erroneous responses generated by the system. Such revised responses should be taken into account when predicting subsequent utterances, which is difficult for concatenation-based approaches such as PICARD.

After obtaining two parts of the input, we tokenized them individually and concatenated the tokens to feed into an LLM. For each token, the LLM produces an embedding vector of d dimensions (768 dimensions using BERT). Therefore, the LLM returns

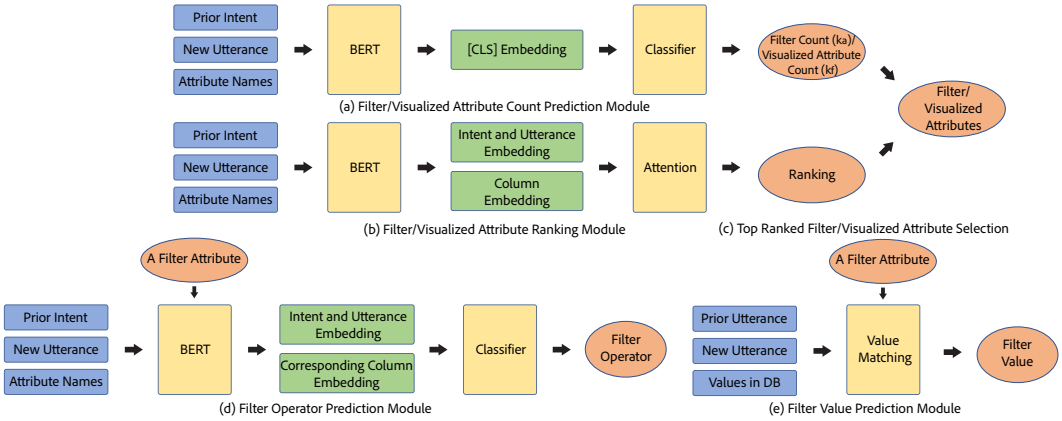

Figure 3: Luna's architecture. (a)-(c) illustrate the modules of predicting visualized attributes and attributes in filters. (d) shows how the operator in a filter is predicted. (e) represent the module for predicting filter values. The blue boxes are model inputs; yellow boxes are models; green boxes are intermediate model outputs; orange boxes are visualization intent predictions.

an embedding matrix with a shape of $L \times d$ for input with length L. Based on the type of content in the input, we split the entire embedding matrix into two parts: *intent and utterance embeddings* and *column embeddings*. Intent and utterance embeddings refer to the matrix consisting of the embeddings of both the most recent intent in the conversation and the current utterance. Similarly, column embeddings refer to column token embeddings. Note that a column may have more than one token when the column name consists of multiple words. To represent each column name with only one embedding vector, we use the embedding corresponding to the column name's first token. We decouple the utterance and column embeddings so that we can use an attention mechanism for the utterance on columns, i.e., to compute how the utterance attends to the columns. Furthermore, the BERT model also returns the embedding of a special token, [CLS], with a pooling scheme [8]. The embedding of the special token summarizes the entire input to BERT, and is frequently used for classifying the entire input. The following sections will introduce how the output embeddings are leveraged in different modules in detail.

4.3 Visualized Attribute Prediction

Luna uses a two-step algorithm to predict the attributes to be visualized. We first predict the number of attributes with a visualized attribute count prediction module (Fig. 3(a)), where we use the BERT model to compute a summary embedding of the context, the current utterance and the columns, i.e., the embedding of the [CLS] token. Then the embedding is fed into a two-layer neural network for classifying the number of attributes, k_a.

Concurrently, we use a visualized attribute ranking module (Fig. 3(b)) to rank all attributes in the dataset according to their relevance to the previous intent and the current utterance. The visualized attribute ranking module consists of another BERT model with a multi-head attention layer followed by a feed-forward fully connected and softmax layer. In the attention layer, the column embeddings act as the query while the utterance embeddings are the keys and values. This design helps select the data attribute(s) to visualize by explicitly forcing the utterance tokens to attend to the column headers. This architecture generates a probability distribution over all columns based on the utterance embeddings' attention on column embeddings. The probability scores indicate how likely the users intend to visualize these columns according to their previous intents and new utterances. All columns are finally ranked from the highest post-softmax probability to the lowest one.

After we obtain the number of attributes (k_a), the top-k_a attributes ranked by the visualized attribute ranking module are predicted as the user's intended attributes for visualization (Fig. 3(c)).

The reason for Luna's two-step strategy with attention mechanism is two-fold. First, the nature of the attention mechanism fits the attribute prediction task well. Its structure explicitly considers the correspondence between tokens in input query, key, and values. Since we aim to understand how the words in users' utterances match the semantically meaningful attribute names, the attention mechanism suits our task well.

Moreover, our approach can handle datasets with many attributes, unlike prior work. We considered using a multi-class multi-target classification head following a BERT model to select the attributes to be visualized. When training the model with such an architecture, we had to fix the number of classes during classification as the maximum number of attributes in the training set. Considering the long-tail nature of dataset sizes revealed in previous work (e.g., [13]), the models might not be trained sufficiently on datasets with large numbers of attributes and might not generalize well. Furthermore, in the inference phase, such an architecture cannot be applied to datasets with more attributes than the number of classes fixed upfront. Compared to leveraging a multi-class multi-target classification head, our approach does not add other constraints to the number of input data attributes despite the input token limit of the BERT model. A quantitative comparison is in Sec. 6.2.

4.4 Filter Prediction

Besides the visualized attributes, Luna also predicts the desired data filters to determine the subset of data to be visualized. For the example in Fig. 1(f), such filters include `year > 2016` and `county = 'Contra Costa'`. As the example shows, to get a complete set of filters, we must infer not just the number of filters, but also the attributes (e.g., year and county), operators (e.g., > and =), and values (e.g., 2016 and Contra Costa) of each filter.

Luna predicts the intended filters using four modules jointly (see Fig. 3). Similar to predicting the visualized attributes, we first use the BERT model followed by a classifier to predict the count of filters, k_f, in the filter count prediction module (Fig. 3(a)). Then, the attributes in the filters are inferred by picking the top-k_f relevant attributes computed by the filter attribute ranking module (Fig. 3(b)), where a BERT model with an attention layer is employed.

After obtaining the data column of each filter, we next predict the operator and value in each filter. For a given filter, the operator and the value are conditioned only on the filter attribute, previous intent(s), and the new utterance. Decoupling the filters from each other simplifies the prediction space of operators and values.

To predict the operator in a filter, we augment the input to the BERT model by including the filter's column name in the input (Fig. 3(d)). The rest of the BERT input is identical to the other modules. After the intent and utterance embeddings are computed, we concatenate them with the predicted column's embedding vector and passed them to a classification head with a two-layer neural network. We do not feed the embedding of the [CLS] token into the classification head as in the filter attribute prediction module, since the embedding of [CLS] also summarizes the embedding of all columns, which may lead to interference between the filter's column name and other column names. The classification head returns the operator in the current filter. The possible operators include <, >, =, and ≠.

Finally, we predict the value used in each filter (Fig. 3(e)). Unlike other modules, the value prediction module is not based on machine learning. In our approach, the current utterance and the prior utterance are first tokenized into individual words. Using these tokens, a set of consecutive tokens with lengths 1 to 5 is generated. We call a sequence of n consecutive tokens as an n-gram. By now, the model has determined the data filter column. For numerical columns, any n-gram that can be parsed as a valid floating point number is added to a set of candidate values. For all other types of columns, all unique values from the dataset for the data filter column are retrieved. Any n-gram that exactly matches such a value is added to the set of candidate values. In most cases, the number of candidate values is exactly one and we are done. However, there may be situations where multiple candidate values exist. Consider the case where two queries are sequentially provided:

(1) Then can you also show the numbers of those cases in *Contra Costa* County? **(1)**

(2) What about those cases in *Alameda* County? **(2)**

The candidate set of values is then sorted temporally and the value resulting from the most recent utterance is selected. The temporal order is determined by the label of each utterance, i.e., the bold number next to each utterance in the examples. We desire the more "recent" match since it is more relevant to the desired intent.

5 Evaluation Setup

We now discuss the experimental setup to evaluate Luna. Luna and our prompt for GPT models are available in our code repository[1].

[1]https://github.com/luna-conversation-vis/luna

5.1 Dataset

We derive a custom dataset from CoSQL [45], a conversational text-to-SQL dataset.

CoSQL. CoSQL consists of 200 SQL databases split across training, validation, and test sets. The training set consists of 2159 conversations and 7343 interactions, while the validation set has 293 conversations and 1007 interactions. Since the test set of CoSQL is not publicly available, we created a test set by sampling 35% interactions of the original validation set. The other 65% interactions were still used as the validation set. When creating the test set, we ensured that the test and validation sets do not share the same databases. Our final test set includes 103 conversations and 349 interactions (with one interaction removed due to malformed SQL), and the validation set has 190 conversations and 657 interactions.

Dataset Processing. We extract visualization intent from the SQL queries in CoSQL to create a dataset for training and testing Luna with other baseline approaches. Fig. 4 shows an example of how we extract visualization intent; Figs. 4(a)-(b) show an original interaction in CoSQL. First, we use a fork of Mozilla's SQL parser called mo-sql-parsing [18] to generate a parse tree of each SQL query (Fig. 4(c)). Using each SQL parse tree, we extract the `SELECT` attributes and predicate clauses. Each predicate clause has the form `ATTRIBUTE, OPERATOR, VALUE` where `OPERATOR` is one of >, ≠, <, =. `ATTRIBUTE` is one of the columns of the SQL database and `VALUE` is a fixed value. For < and >, `VALUE` must be a numerical value that may or may not be in the SQL database, but when `OPERATOR` is = or ≠, `VALUE` may be a numerical value or string. The `SELECT` attributes and predicate clauses are mapped to visualized attributes and data filters in visualization intent (Fig. 4(d)).

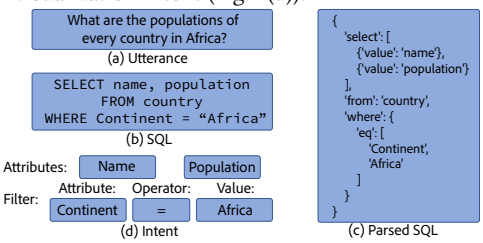

Figure 4: Example CoSQL interaction being transformed.

5.2 Luna Setup

Module setup. For Luna's visualized attribute count and filter count prediction modules, the hidden layer has size 1028. We set the number of classes in the attribute and predicate count modules to 7 (i.e., 0 – 6 attributes) and 5 (i.e., 0 – 4 filters), respectively, based on the training data. In our modules with attention mechanisms (i.e., attribute ranking and predicate ranking prediction), the number of attention heads is set to 8 with 64 dimensions. In the filter operator prediction module, the size of the hidden layer is 512. The number of classes in the predicate operator is set to 4 (i.e., <, >, =, and ≠).

Input formulation. As in Sec. 4.2, the input to our model includes the previous intent when there are prior interactions. We feed the ground truth previous intent to Luna together with the new utterance to predict the current intent. To understand Luna's ability to handle cascading errors from a wrongly predicted intent, we also conduct an experiment where we feed Luna's previously predicted intent instead of the ground truth. The accuracy of Luna with this

setting is also provided for reference in Table 2. We note, however, that users will typically fix any errors in the intent rather than let the errors cascade (i.e., making sure they see what they intended before proceeding). Therefore, when not mentioned, we report the accuracy of using ground truth as the input by default.

Training details. We use 8 training epochs to fine-tune BERT and train our classifiers using the training set. Then the best model is selected with the validation set. To train the model, we use the Adam optimizer [17] with a BERT learning rate of $5e^{-5}$ and a head learning rate of $5e^{-5}$. Furthermore, we noticed an unbalanced data distribution in all classification sub-tasks (i.e., visualized attribute count, filter count, and filter operator prediction). To mitigate data skew, we over-sample the training set by categories (i.e., the possible numbers of visualized attributes, the possible numbers of filters, and filter operator types).

5.3 Baseline Models Setup

We compare the performance of Luna with six other state-of-the-art approaches. We used the publicly available version of the pre-trained models to run the experiments.

NL4DV. As far as we know, NL4DV [26, 28] is the only publicly available system that creates visualizations based on multi-turn conversations. Notably, NL4DV can recommend multiple visualizations with ranking. To make the results comparable, we keep the visualization with the highest rank.

PICARD. We tried to find other models by scanning through the CoSQL leaderboard [1] and noticed that the top 5 models are not publicly accessible or can only predict the data columns and operators in data filters without detailed values to be filtered (e.g., STAR [6]), hence we chose the 6th-ranking model (as of May 2024), PICARD [33], as a representative of the best state-of-the-art conversational NL2SQL models. We use two PICARD variants, one with with a T5 model with 3 billion parameters (denoted as T5-3B) [31], and a smaller T5 model with 880 million parameters (denoted as T5-Large) since the number of parameters is closer to the total number of parameters in Luna. For brevity, two PICARD variants are denoted as PICARD-3B and PICARD-Large.

CD-Seq2Seq. Following prior work [14, 48], we also include the results of CD-Seq2Seq (short for Context-dependent Seq2Seq) that is used as a baseline approach for the CoSQL dataset [45]. However, CD-Seq2Seq is unable to predict the values in predicates. Therefore, we consider all predicate values correct when reporting the performance, which is a loose upper bound on its actual accuracy.

GPT models. GPT-3.5 and GPT-4 have shown good performance in responding to natural language queries with visualizations [9, 25]. However, previous studies leveraging GPT models focus on answering one-shot natural language queries with visualizations. Furthermore, such models return visualization code instead of visualization intent, which is different from Luna's output. Therefore, to compare Luna with GPT models, we followed prior work [4, 9, 25] and prompt GPT-3.5-turbo-0613 and GPT-4-0613 through Azure OpenAI APIs to infer visualization intent and return it in JSON.

6 Evaluation Results

We compare Luna's performance with baseline approaches in Sec. 5.3 and explore alternatives in the architecture of Luna.

6.1 Comparison with Baseline Approaches

We evaluate the aforementioned approaches on their ability to correctly predict visualization intent.

Table 2: Comparison of existing approaches with the highest accuracies in bold.

Model	Validation Accuracy	Test Accuracy
Luna (Correct Previous Intent)	**51.29%**	**57.31%**
Luna (Cascade)	46.73%	54.57%
NL4DV	10.48%	16.31%
PICARD-3B	38.97%	49.57%
PICARD-Large	38.66%	50.14%
GPT-3.5	34.15%	42.57%
GPT-4	33.03%	44.87%
CD-Seq2Seq	18.57%	26.65%

6.1.1 Results. Table 2 shows the overall accuracy. Luna achieves the best performance among all four tested approaches, with 51.29% and 57.31% accuracy on the validation and the test set respectively, when using the ground truth previous intent as part of the input (Luna (Correct Previous Intent)). When we feed the previously predicted intent to Luna without re-training (Luna (Cascade)) the accuracy scores are 46.73% and 54.57% on the validation and the test sets, respectively. The small difference between the results reveals that our model can be affected by cascading errors, but the performance loss is minor. Since users will typically correct such errors, as discussed in Sec. 5, and since the difference is minor, we focus on the former in the following. Overall, these results indicate that Luna has a much greater ability to infer visualization intent compared to other approaches.

Compared to NL4DV, the latest conversational NL2VIS approach, Luna has a substantial increase of 251.01% in test accuracy. It is also superior to NL2SQL approaches, including PICARD variants and CD-Seq2Seq by 15.61%, 14.30%, and 115.05%. Note that the improvement relative to CD-Seq2Seq is actually even greater, due to our assumption that CD-Seq2Seq can predict all of the filter values correctly (see Sec. 5), and therefore its reported accuracy is an overestimate. Notably, though the backbone of Luna is BERT, a relatively outdated and small LLM, Luna still outperforms the latest and much larger LLMs, GPT-3.5 and GPT-4, by 32.28% and 27.72%, respectively. Since NL4DV and CD-Seq2Seq perform worse than Luna, the PICARD variants, and GPT models, we focus on comparing Luna with PICARD and GPT in the following.

We conclude with two observations for Luna's performance comparison against PICARD and GPT. First, Luna shows better performance in both the accuracy scores of visualized attributes and filters. To understand the performance difference between these approaches, we inspect their accuracy breakdown on correctly identifying the visualized attributes and filters. The results in Table 4 show that Luna has a better performance on both components of visualization intent, verifying the effectiveness of Luna's approach in dividing visualization intent inference into sub-tasks and training specialized models for specific tasks.

To further understand the differences, Table 3 compares Luna, GPT, and PICARD in five cases: correct attributes and filters, correct attributes but wrong filters, correct filters but wrong attributes, wrong attributes and filters, and other errors (such as JSON structure errors when evaluating GPT models and unexecutable errors when evaluating PICARD models). One important problem with

Table 3: Accuracy breakdown of Luna, PICARD, and GPT across different categories. A: attributes, F: filters.

Model - Dataset Split	Correct A+F	Correct A	Correct F	Wrong A+F	Others
Luna- Val	51.29%	22.83%	19.18%	6.70%	N/A
PICARD-3B - Val	38.97%	9.59%	8.83%	2.44%	40.18%
PICARD-Large - Val	38.66%	8.83%	8.83%	2.74%	40.94%
GPT-3.5 - Val	34.15%	25.15%	19.36%	16.01%	5.34%
GPT-4 - Val	33.03%	19.42%	29.05%	17.43%	1.07%
Luna- Test	57.31%	19.77%	14.90%	8.02%	N/A
PICARD-3B - Test	49.57%	8.88%	10.03%	2.01%	29.51%
PICARD-Large - Test	50.14%	5.44%	9.17%	2.58%	32.66%
GPT-3.5 - Test	42.57%	23.91%	16.91%	12.83%	3.79%
GPT-4 - Test	44.87%	17.01%	28.15%	8.80%	1.17%

Table 4: Comparison of existing end-to-end methods broken down by visualized attributes and data filters. The highest accuracies are in bold. A: attributes, F: filters.

Model	Val-A	Val-F	Test-A	Test-F
Luna	**74.12%**	**70.47%**	**77.08%**	72.21%
NL4DV	23.79%	16.64%	31.69%	23.08%
PICARD-3B	48.55%	47.79%	58.45%	59.60%
PICARD-Large	47.49%	47.49%	55.59%	59.31%
GPT-3.5	59.30%	53.51%	66.47%	59.48%
GPT-4	52.45%	62.08%	61.88%	**73.02%**
CD-Seq2Seq	25.11%	30.90%	35.53%	38.11%

the PICARD variants is that they often generate queries that are not executable. Among all generated SQL queries, we notice that 40.18% and 29.51% of the queries in the validation and the test sets for PICARD with T5-3B are not executable while the numbers are 40.94% and 32.66% for PICARD with T5-Large, respectively. The main reasons for these failures include syntax errors and semantic errors, such as those in Fig. 1. Some frequent syntax errors include the misuse of quotation marks (see Fig. 1(f)) and the incorrect selection when using `group by`, such as the queries in Fig. 1 (c) and (d). The results imply that the structure or the grammar of the output can be a severe issue for the end-to-end generative models that have to learn them from training data. Meanwhile, most of the semantic errors pertain to data types, like applying `avg` on text columns. These errors show that it is unreliable to generate visualization intent from only semantic information in the natural language utterances. Such issues also lower the performance of the GPT models. They can generate results without inappropriate JSON, such as invalid structure, missing keys, or natural language responses. For example, GPT-4 returns *"Sorry, but this dataset doesn't contain any information about Gonzalo Higuaín"* when the query would like to filter a customer whose name is Gonzalo Higuaín.

To conclude, PICARD often fails due to the lack of ability to generate executable SQL queries but shows potential in comprehending the intent from conversations correctly. In contrast, GPT models, especially GPT-4, are often unable correctly understand visualization intent from conversations but maintain the correctness of output structure, possibly because they aren't specifically trained for identifying visualization intent. Luna combines their advantages: it understands visualization intent through training on conversational intent and guarantees an always executable output following our task breakdown.

Finally, Luna is smaller than both T5-3B and T5-Large PICARD variants in terms of model size by 81.67% and 37.50%. The smaller size allows Luna to run on resource-limited devices with uncompromised performance. Following Shen *et al.* [36], we compare the peak GPU memory usage and average time of inference of Luna and PICARD with an Nvidia A100-40GB. The batch size of inference is set to 1. The memory usage and time are computed over both the validation and the test set. The memory usage data is collected with Nvidia System Management Interface (SMI) [29] per second. Luna achieves better performance (with at most 2,843 MB memory) using around 12% memory of PICARD with T5-3B (with at most 23,704 MB memory) and 37% memory of PICARD with T5-Large (with at most 7,751 MB memory) and also takes much less time for inference (Average inference time: Luna: 0.1 Sec, PICARD with T5-3B: 142.7 Sec[2], PICARD with T5-Large: 152.6 Sec). Moreover, since Luna consists of several BERT models, it is possible to use multiple devices to train and deploy Luna in a distributed manner, which enhances the flexibility of Luna on resource-limited devices.

Table 5: Individual component accuracies and alternatives. The components in *italic* were applied in Luna. The accuracy scores in bold were the highest ones.

Component	Method	Val	Test
Visualized Attribute Count	*Classification*	**89.65%**	**92.55%**
	Attention (English e.g., "one")	83.71%	87.97%
	Attention (Numerical e.g.,"1")	86.76%	91.69%
Visualized Attribute	*Attention*	**80.37%**	**81.66%**
	Classification	75.65%	78.80%
Filter Count	*Classification*	**86.00%**	**85.39%**
	Attention (English e.g., "one")	81.28%	81.38%
	Attention (Numerical e.g.,"1")	81.28%	81.95%
Filter Attribute	*Attention*	**86.30%**	**86.82%**
	Classification	80.21%	82.24%
Filter Operator	*Classification*	**99.09%**	97.13%
	Attention (English e.g., "less than")	97.87%	**97.42%**
	Attention (Symbolic e.g., "<")	98.94%	**97.42%**
Filter Value	*Temporal Text Matching (Last)*	**87.06%**	**85.39%**
	Temporal Text Matching (First)	81.89%	80.23%

6.2 Individual Module Evaluation

We now report the performance of individual Luna modules compared to alternative designs discussed in Secs. 4.3 and 4.4. For each component with BERT, we compare alternative model choices, i.e., classification heads or attention-based heads as introduced in Sec. 4. For filter value prediction, two strategies of temporal value matching, i.e., picking the first or last matched value, are compared. Since some modules may rely on the input of the previous module (e.g., filter operator prediction), we feed the ground truth to these modules to remove cascading errors. The results are shown in Table 5.

Visualized Attribute and Filter Count Prediction. As shown in Fig. 3(a), we apply a classifier to predict the number of visualized and filter attributes. An alternative design is to use an attention-based head that attends the embedding of the previous intent, new utterance, and the attributes to the number of attributes. The model structure of this alternative design is similar to the one shown in Fig. 3(b). The representation of the number of attributes can be either English words (e.g., "one", "two") or a string version of the numbers (e.g., "1", "2"). Since there is no apparent difference

[2]According to the PICARD paper [33], their inference time on Spider [46] is 3.1 Sec. We both apply Nvidia A100 with 40GB of memory to run PICARD variants.

between the two representation approaches, we train and evaluate the model with both using the same training settings in Sec. 5.2.

The results in Table 5 show that the classification head achieves the best accuracy among three alternative choices, achieving 89.65% and 92.55% when predicting the number of visualized attributes on our validation and test sets. The accuracy of predicting filter counts is 86.00% and 85.39% on the two sets, respectively. We believe the reason for the lower accuracy scores of the attention mechanism is that training the attention mechanism requires the model to learn the semantic relationship between the embedding of input and the embedding of numbers.

Visualized and Filter Attribute Prediction. After obtaining the number of visualized attributes and filters, we select the top-k attributes ranked by the ranking model (Fig. 3(b)). Since there is no ground truth for the rankings of attributes, we cannot evaluate the attribute ranking models directly. Instead, we use the ground truth numbers of visualized attributes and filters to select the visualized and filter attributes. Then, the selected attributes are compared with the ground truth to evaluate the ranking models indirectly.

As in Sec. 4.3, one alternative to the attention mechanism is a multi-target multi-class classifier. We implement a two-layer fully-connected neural network with a hidden size of 1028. The number of output classes is set to 132 classes, representing column choices. 132 is the maximum number of columns in any database across the training, validation, and test sets. In order for the classifier to be applicable to every database, it must have at least 132 classes.

As our results show, the classification heads perform worse than the attention-based head in both tasks. Each database has a different set of column names, with different semantic meanings that make the classification task challenging. Most classification tasks keep the meaning of each class consistent to ensure that predictions are standardized across inputs. This assumption does not hold in our setting where each class has a different meaning depending on the input database schema. Furthermore, as Sec. 4.3 mentions, the large number of classes, i.e., 132, can bring additional challenges for training a classifier with satisfactory performance.

Predicate Operator Selection. In Luna, a classifier is used to predict the operators in the filters. Instead of a classification head, we may also apply an attention mechanism that attends to either the symbolic version of each operator (i.e., <, >, =, !=) or the English description (i.e., less than, greater than, equal to, not equal to). As shown in Table 5, these three alternative choices work almost equally well. They all achieve an accuracy of over 97% on both the validation and the test set. We hypothesize that the task of predicting filter operators is relatively simple since the filter operations can be directly included in users' utterances. For example, the word "after" in "What about those cases happened after 2016?" directly indicates that the operator should be "larger than." Therefore, all three model choices can handle the task well. Based on the validation accuracy, we selected the classification head to conduct the operator prediction task in Luna.

Filter Value Prediction. As mentioned in Sec. 4.4, we adopt a simple text matching-based approach to identify the filter values. There are two possible designs when searching for the matched text. We can either select the first matched value or the last matched one. The experimental results in Table 5 justify our design rationale that the last matched value can be more relevant to the user's intent.

7 Related Work

Natural Language to Visualization. Recent work has explored natural language-to-visualization (NL2VIS) interfaces [35] supporting either multi-turn conversations or only single questions. Early systems, e.g., [10, 34, 37, 39, 44], adopt a rule-based approach to parse user utterances and match words with data columns or operations, greatly limiting their flexibility. Some work applies LLMs to understand single-shot natural language utterances and recommend visualizations. Early work [7, 21, 22] trained new models to address the NL2Vis challenges while recent work [9, 25] leverage general-purpose LLMs. We find that Luna outperforms state-of-the-art LLMs such as GPT-3.5 and GPT-4.

Previous research has shown that visual data analysis is progressive [16, 19, 41]. However, the aforementioned approaches does not support such a progressive workflow. A few prior tools have explored conversational NL2Vis using rule-based approaches, such as [12, 38]. Finally, the recent version of NL4DV [26] extends the original version [28] to a conversational setting by searching the user input for keywords to indicate a follow-up utterance. Due to their rule-based nature, these approaches easily break.

Natural Language to SQL. Most conversational NL2SQL approaches apply end-to-end generative models to translate natural language utterances into SQL. For example, CD-Seq2Seq [40, 46] applies bi-directional recurrent neural networks to encode the context and the current utterance and decode the embedding to SQL queries. R^2SQL [14] and EditSQL [48] enhances the encoder with BERT-based models [8]. More recently, PICARD [33] and UnifiedSKG [43] leverage T5 [31]. There are two challenges when applying these end-to-end NL2SQL models in inferring visualization intent. First, it is hard to guarantee that the generated SQL queries are both syntactically and semantically correct [33]. Furthermore, some complex SQL queries cannot be easily interpreted as visualization intent, such as those with subqueries. In contrast, Luna guarantees the generation of correct and interpretable visualization intent.

8 Conclusion and Future Work

We propose Luna as a framework to infer visualization intent from conversation. Unlike end-to-end generative models, Luna leverages several specialized LLM-based modules to predict different aspects of visualization intent separately. We demonstrate the effectiveness of Luna through a quantitative comparison with baseline NL2Vis and NL2SQL approaches. We also validate the efficacy of Luna in a real world scenario of exploring police misconduct. As future work, we plan to enhance Luna to identify visualization-specific aspects, such as visualization type. We will also explore interface modalities for easily fixing incorrect intent. Furthermore, evaluating Luna with additional datasets and user studies can reveal more insights.

Acknowledgements. This work is supported in part by the NSF through grants DGE-2243822, IIS-2129008, IIS-1940759, IIS-1940757, IIS-1955488, IIS-2027575, ARO W911NF2110339, ONR N00014-21-1-2724, and DOE award DE-SC0016260, DE-SC0021982, by the Hong Kong RGC GRF through the grant 16210321, funds from the State of California, as well as EPIC lab sponsors: G-Research, Adobe, Microsoft, Google, and Sigma Computing. We also thank our collaborators: David Barstow, Tristan Chambers, Lisa Pickoff-White, Cheryl Phillips, and Tarak Shah, for their help and encouragement.

References

[1] [n. d.]. CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. https://yale-lily.github.io/cosql.

[2] [n. d.]. GPT-3.5. https://platform.openai.com/docs/models/gpt-3-5.

[3] [n. d.]. PowerBI. https://www.microsoft.com/en-us/power-platform/products/power-bi.

[4] [n. d.]. Prompt engineering. https://platform.openai.com/docs/guides/prompt-engineering.

[5] [n. d.]. Tableau. https://www.tableau.com/.

[6] Zefeng Cai, Xiangyu Li, Binyuan Hui, Min Yang, Bowen Li, Binhua Li, Zheng Cao, Weijie Li, Fei Huang, Luo Si, and Yongbin Li. 2022. STAR: SQL Guided Pre-Training for Context-dependent Text-to-SQL Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 1235–1247. https://aclanthology.org/2022.findings-emnlp.89

[7] Xinyun Chen, Linyuan Gong, Alvin Cheung, and Dawn Song. 2021. PlotCoder: Hierarchical Decoding for Synthesizing Visualization Code in Programmatic Context. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers)*. Association for Computational Linguistics, 2169–2181.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[9] Victor Dibia. 2023. LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models. *arXiv preprint arXiv:2303.02927* (2023).

[10] 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*, Celine Latulipe, Bjoern Hartmann, and Tovi Grossman (Eds.). ACM, 489–500. https://doi.org/10.1145/2807442.2807478

[11] Soreath Hok, Molly Peterson, and Lisa Pickoff-White. 2022. Bakersfield Police Department fails to identify people in crisis, thwarting reform. https://www.kvpr.org/local-news/2022-04-12/bakersfield-police-department-fails-to-identify-people-in-crisis-thwarting-reform

[12] 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

[13] Kevin Zeng Hu, Snehalkumar (Neil) S. Gaikwad, Madelon Hulsebos, Michiel A. Bakker, Emanuel Zgraggen, César A. Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çagatay Demiralp. 2019. VizNet: Towards A Large-Scale Visualization Learning and Benchmarking Repository. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 662. https://doi.org/10.1145/3290605.3300892

[14] Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic Hybrid Relation Exploration Network for Cross-Domain Context-Dependent Semantic Parsing. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. AAAI Press, 13116–13124.

[15] John D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[16] Mary Beth Kery and Brad A. Myers. 2017. Exploring exploratory programming. In *Proceedings of the 2017 IEEE Symposium on Visual Languages and Human-Centric Computing*, Austin Z. Henley, Peter Rogers, and Anita Sarma (Eds.). IEEE Computer Society, 25–29. https://doi.org/10.1109/VLHCC.2017.8103446

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

[18] Kyle Lahnakoski. [n. d.]. More SQL Parsing! https://github.com/klahnakoski/mo-sql-parsing

[19] Doris Jung Lin Lee, Himel Dev, Huizi Hu, Hazem Elmeleegy, and Aditya G. Parameswaran. 2019. Avoiding Drill-down Fallacies with VisPilot: Assisted Exploration of Data Subsets. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. ACM, 186–196. https://doi.org/10.1145/3301275.3302307

[20] Doris Jung Lin Lee, Dixin Tang, Kunal Agarwal, Thyne Boonmark, Caitlyn Chen, Jake Kang, Ujjaini Mukhopadhyay, Jerry Song, Micah Yong, Marti A. Hearst, and Aditya G. Parameswaran. 2021. Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows. *Proceedings of the VLDB Endowment* 15, 3 (2021), 727–738. https://doi.org/10.14778/3494124.3494151

[21] Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data. In *Proceedings of the 14th IEEE Pacific Visualization Symposium*. IEEE, 11–20. https://doi.org/10.1109/PacificVis52677.2021.00010

[22] 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 (2022), 217–226. https://doi.org/10.1109/TVCG.2021.3114848

[23] Jock D. Mackinlay. 1986. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics* 5, 2 (1986), 110–141. https://doi.org/10.1145/22949.22950

[24] Jock D. Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show Me: Automatic Presentation for Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1137–1144. https://doi.org/10.1109/TVCG.2007.70594

[25] Paula Maddigan and Teo Susnjak. 2023. Chat2VIS: Generating Data Visualizations via Natural Language Using ChatGPT, Codex and GPT-3 Large Language Models. *IEEE Access* 11 (2023), 45181–45193. https://doi.org/10.1109/ACCESS.2023.3274199

[26] Rishab Mitra, Arpit Narechania, Alex Endert, and John Stasko. 2022. Facilitating Conversational Interaction in Natural Language Interfaces for Visualization. , 6-10 pages. https://doi.org/10.1109/VIS54862.2022.00010

[27] Dominik Moritz, Chenglong Wang, Greg L. Nelson, Halden Lin, Adam M. Smith, Bill Howe, and Jeffrey Heer. 2019. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 438–448. https://doi.org/10.1109/TVCG.2018.2865240

[28] Arpit Narechania, Arjun Srinivasan, and John T. 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

[29] Nvidia. [n. d.]. System Management Interface SMI. https://developer.nvidia.com/nvidia-system-management-interface

[30] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 140:1–140:67.

[32] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 341–350. https://doi.org/10.1109/TVCG.2016.2599030

[33] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 9895–9901. https://doi.org/10.18653/v1/2021.emnlp-main.779

[34] 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*. ACM, 365–377. https://doi.org/10.1145/2984511.2984588

[35] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2022. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2022), 3121–3144.

[36] Yikang Shen, Zheyu Zhang, Tianyou Cao, Shawn Tan, Zhenfang Chen, and Chuang Gan. 2023. ModuleFormer: Learning Modular Large Language Models From Uncurated Data. *CoRR* abs/2306.04640 (2023). https://doi.org/10.48550/arXiv.2306.04640

[37] Tarique Siddiqui, Paul Luh, Zesheng Wang, Karrie Karahalios, and Aditya Parameswaran. 2020. Shapesearch: A flexible and efficient system for shape-based exploration of trendlines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 51–65.

[38] Arjun Srinivasan and Vidya Setlur. 2021. Snowy: Recommending Utterances for Conversational Visual Analysis. In *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology*. ACM, 864–880. https://doi.org/10.1145/3472749.3474792

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

[40] Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to Map Context-Dependent Sentences to Executable Formal Queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2238–2249.

[41] John W. Tukey. 1977. *Exploratory Data Analysis*. Addison-Wesley.

[42] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock D. Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Towards a General-purpose Query Language for Visualization Recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM, 4. https://doi.org/10.1145/2939502.2939506

[43] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir

Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 602–631. https://aclanthology.org/2022.emnlp-main.39

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

[45] Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. 2019. CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 1962–1979. https://doi.org/10.18653/v1/D19-1204

[46] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 3911–3921. https://doi.org/10.18653/v1/d18-1425

[47] Zehua Zeng, Phoebe Moh, Fan Du, Jane Hoffswell, Tak Yeon Lee, Sana Malik, Eunyee Koh, and Leilani Battle. 2022. An Evaluation-Focused Framework for Visualization Recommendation Algorithms. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 346–356. https://doi.org/10.1109/TVCG.2021.3114814

[48] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 5338–5349. https://doi.org/10.18653/v1/D19-1537