Towards Robustness of Text-to-Visualization Translation against Lexical and Phrasal Variability

Anonymous ACL submission

Abstract

Text-to-Visualization (text-to-vis) is an emerging task in the natural language processing (NLP) area that aims to automatically generate data visualizations from natural language questions (NLQs). Despite their progress, existing text-to-vis models often heavily rely on lexical matching between words in the questions and tokens in data schemas. This overreliance on lexical matching may lead to a diminished level of model robustness against input variations. In this study, we thoroughly examine the robustness of current text-to-vis models, an area that has not previously been explored. In particular, we construct the first robustness dataset **nvBench-Rob**, which contains diverse lexical and phrasal variations based on the original text-to-vis benchmark nvBench. Then, we found that the performance of existing text-to-vis models on this new dataset dramatically drops, implying that these methods exhibit inadequate robustness overall. Finally, we propose a novel framework based on Retrieval-Augmented Generation (RAG) technique, named **GRED**, specifically designed to address input perturbations in these two variants. The framework consists of three parts: NLQ-Retrieval Generator, Visualization Query-Retrieval Retuner and Annotation-based Debugger, which are used to tackle the challenges posed by natural language variants, programming style differences and data schema variants, respectively. Extensive experimental evaluations show that, compared to the state-of-the-art model RGVisNet in the Text-to-Vis field, GRED performs better in terms of model robustness, with a 32% increase in accuracy on the proposed nvBench-Rob dataset.[1]

1 Introduction

Data visualization (DV) has emerged as an indispensable tool in the industry for extracting insights from massive data. It surpasses verbal expressions, offering a clear and effective presentation of insights derived from raw data. The process of creating DVs involves programming declarative visualization languages (DVLs) to select relevant data and determine how to present it. With a wide variety of different DVLs available—each characterized by its own distinctive grammar and syntax, such as Vega-Lite (Satyanarayan et al., 2018), ggplot2 (Gómez-Rubio, 2017), ZQL (Siddiqui et al., 2016), and ECharts (Li et al., 2018)—the need for considerable domain knowledge and proficiency in DVL is required, posing a particularly challenge to those who lack technical expertise.

To enhance the accessibility of DV, a task named text-to-visualization (text-to-vis) has been proposed, which offers a mechanism to automatically transform natural language questions (NLQs) into DV charts. As shown in Figure 1, the text-to-vis system requires users to simply ask an NLQ, such as, *"Draw a bar chart about the change of salary over hire_date, sort x axis in asc order."* It then automatically generates the final DV, such as a bar chart, by interfacing with the database, thereby circumventing the need for users to code directly in a DVL.

To deploy text-to-vis models in real-life, it is crucial for these models to possess the capability to handle NLQs from diverse users. Therefore, the *robustness* of the model plays an important role in evaluating the performance of text-to-vis models. High model performance requires robust performance on noisy inputs. However, the robustness of text-to-vis models poses a significant challenge. In our analysis (Section 3), we found that even small perturbations in the input may significantly reduce the performance of existing text-to-vis models. Furthermore, there is still a lack of dedicated robustness datasets and studies in the field to effectively evaluate the robustness of text-to-vis models.

We notice that the NLQs in the original text-

[1] Our code and data are available at https://1drv.ms/f/s!AkYKmrrFYuiAkWnlc5HTJAcWZcUQ?e=9IVLNR.

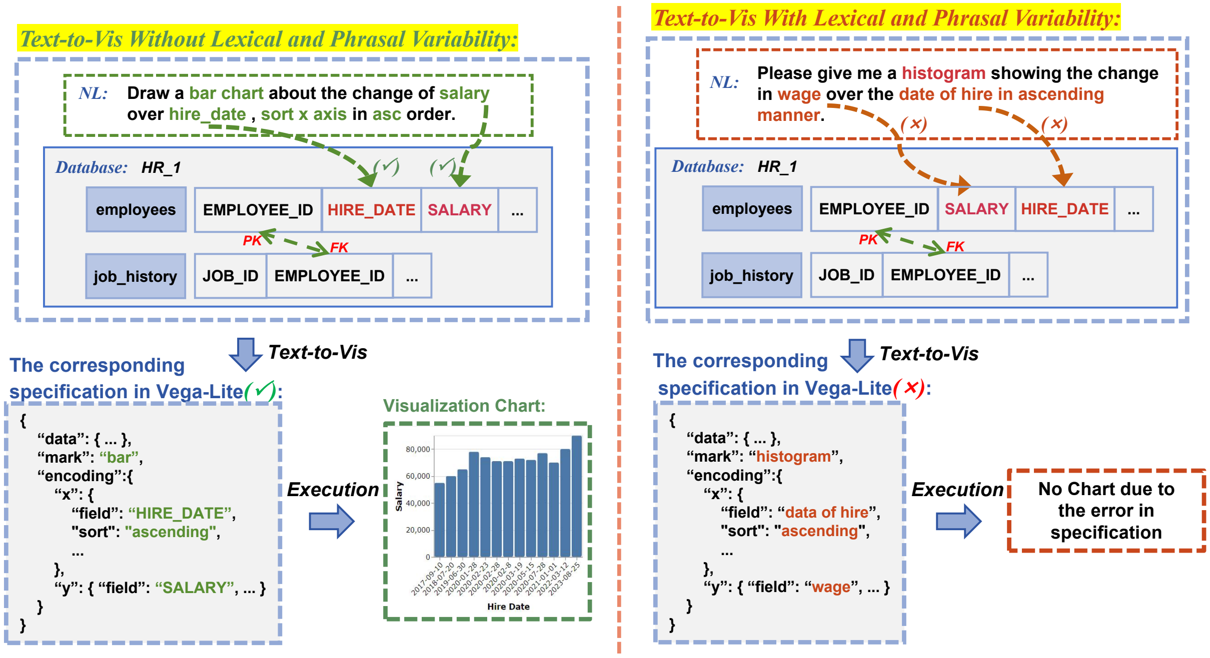


Figure 1: (a) Text-to-vis is dedicated to converting natural language questions (NLQs) into data visualizations (DVs). The current approach heavily relies on explicit matching between words within the NLQs and the table schema. (b) The robustness of existing text-to-vis methods is limited. When small variations in NLQs and table schemas appear, the text-to-vis model fails to generate correct outputs (marked with '×' in red color).

to-vis dataset nvBench (Luo et al., 2021a) usually explicitly mention the information present in the database, like explicit mentions of column names. This characteristic makes the test results of nvBench unsuitable for evaluating the robustness of the text-to-vis models. It is difficult to ascertain whether the model simply memorizes the explicitly mentioned schema, such as column names, or if it genuinely learns the natural mapping relationship between the NLQ and data schema.

The lack of large-scale datasets is one of the significant factors that limits the robustness studies in the text-to-vis field. In this work, we propose the first comprehensive robustness dataset named **nvBench-Rob** to evaluate the robustness of the text-to-vis models. nvBench-Rob aims to provide a comprehensive evaluation of models based on two variants: NLQ and data schema, as shown in Figure 1. With these two variants, we thoroughly examine the robustness of the current text-to-vis models, an area that has not previously been explored. We found that the performance of existing text-to-vis models dramatically drop, implying these methods exhibit inadequate robustness.

To enhance the robustness of text-to-vis models, we propose a novel framework named **GRED** based on the Retrieval-Augmented Generation (RAG)-based technique for Large Language Models (LLMs) (Roziere et al., 2023; Touvron et al., 2023; Gunasekar et al., 2023; Anil et al., 2023; OpenAI., 2024). This framework comprises three core components: NLQ-Retrieval Generator, DVQ-Retrieval Retuner, and Annotation-based Debugger, aimed at addressing variants of NLQs, differences in programming styles, and changes in data schema, respectively. [2]

Specifically, in the preparation phase, GRED utilizes a pre-trained text embedding model (Reimers and Gurevych, 2020; Feng et al., 2020) to convert all NLQs and DVQs contained in the nvBench training set into embedding vectors, thus creating an embedding vector repository. Then, ChatGPT is used to generate natural language annotations for each database, creating a collection of annotated database sets. Once ready, for NLQs sent into the text-to-vis system, GRED first uses the pre-trained text embedding model to convert them into embedding vectors and calculates their cosine similarity with the embedding vectors of NLQs in the training set. Then, the top-K most similar NLQs are selected, and their corresponding examples are combined into a generation prompt in descending order of similarity, which is input into ChatGPT to generate the corresponding DVQ, referred to as

[2]DVQ refers to Data Visualization Query (Luo et al., 2021a; Song et al., 2022), which is a widely-used intermediate representation that connects NLQ with the DVLs like Vega-Lite and ECharts.

DVQ_{gen}. Next, DVQ_{gen} is converted into embedding vectors, and its cosine similarity with DVQ embedding vectors in the library is calculated. The top-K most similar DVQs are selected to construct a tuning prompt, which is then input into ChatGPT to mimic a similar programming style, resulting in DVQ_{rtn}. Finally, the database with natural language annotations and DVQ_{rtn} are combined into a debugging prompt, inputted into ChatGPT to replace inappropriate data schema in DVQ_{rtn}, obtaining the final DVQ_{dbg}.

Experimental results on nvBench-Rob indicate that GRED significantly surpasses existing text-to-vis models in terms of model robustness. Compared to the current state-of-the-art (SOTA) text-to-vis model RGVisNet, GRED achieves an accuracy improvement of over 20% on the single-variant test set and over 30% on the dual-variant test set. These results verify the effectiveness of GRED in enhancing the robustness of text-to-vis models.

In a nutshell, the contributions of our work are threefold:

- To our knowledge, we are the first to comprehensively study the robustness of the text-to-vis task; We hope this work will inspire more research on improving the robust data visualization models.
- We construct nvBench-Rob, the first dedicated dataset to evaluate the robustness of text-to-vis models. We observed significant performance drops of SOTA text-to-vis models on this robustness scenario, revealing that even SOTA models still possess significant potential for further exploration.
- We designed a novel framework called GRED, based on RAG technique. This framework effectively addresses the high sensitivity of text-to-vis models to input perturbations and inconsistencies in programming styles. It provides an innovative paradigm for leveraging Large LLMs to tackle robustness issues in the text-to-vis field.

2 Robustness Dataset: nvBench-Rob

2.1 Overview

We constructed nvBench-Rob benchmark, the first comprehensive robustness evaluation dataset in the field of text-to-vis, through a collaboration between LLMs and humans. Specifically, we utilized LLMs to first modify the original dataset and then manually corrected the modified dataset, which not only saved labor costs but also allowed for diverse language styles and database naming habits within the dataset.

VIS Types	No. of (NL, Vis)
Bar Chart	891
Pie Chart	88
Line Chart	51
Scatter Chart	48
Stacked Bar	60
Grouping Line	11
Grouping Scatter	33
All Types	1182

Hardness	No. of (NL, Vis)
Easy	286
Medium	475
Hard	282
Extra Hard	139
Total	1182

Database	Table	Avg.
104	552	5.31

Table	Column	Avg.
552	3050	5.53

Figure 2: Statistics of the nvBench-Rob Dataset

In nvBench-Rob, we have meticulously designed three robustness test sets to comprehensively evaluate the models from various perspectives: robustness to NLQs, robustness to table schemas, and robustness to the combination of both.

In this section, we will present a detailed overview of our dataset construction method and perform a thorough analysis of the features of nvBench-Rob.

2.2 ChatGPT Modification

The LLM is a kind of large-scale models trained on a massive corpus, demonstrating outstanding capability in natural language processing (NLP) tasks. ChatGPT is one of these representative models. Through ChatGPT (OpenAI., 2024), we can harness its powerful NLP capability to process the dataset.

The existing nvBench dataset usually explicitly mentions table schema (such as column names) and DVQ keywords (e.g., Bin and Group) in the NLQs. This makes it difficult for models trained on this dataset to perform well in scenarios where users have limited knowledge of DV. For instance, users may lack knowledge of table schemas and DVQ syntax (Figure 1). During training, the model may only learn the explicit alignment between NLQ, table schemas, and DVQ, rather than truly understanding how to conduct schema linking semantically. This also reflects that nvBench cannot effectively evaluate the robustness of the model.

LLMs can be potentially used to address the above issues. With its powerful NLU capability, we can utilize LLMs like ChatGPT to simulate various user interaction behaviors, thereby enhancing the robustness of the text-to-vis dataset.

NLQ Reconstruction. We reconstructed the NLQs in nvBench using ChatGPT, without focusing on explicit mentions of table schema and DVQ

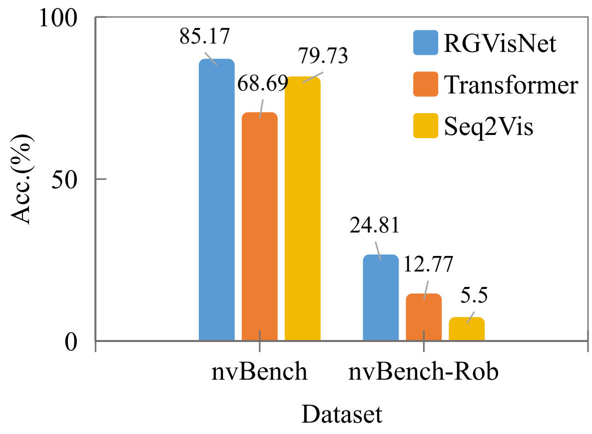


Figure 3: The performance of existing text-to-vis models dramatically **drops** on the nvBench-Rob datasets.

keywords within the sentences. Specifically, we replaced most of the nouns in the sentences with synonyms based on the context, aiming to minimize the explicit mention of table schema in the NLQs. With these modifications, we simulated the interaction between a user who is unfamiliar with both the database information and DVQ syntax and the text-to-vis model.

Schema Synonymous Substitution. We attempted to utilize the approach used in MultiSpider (Dou et al., 2023) by inputting the format "*table(column)[type]*" into ChatGPT, with the aim of having it to return a column name with equivalent meaning in that context. However, the results were consistently unsatisfactory. As a result, we refined the method by constructing prompts that included database name, table names, column names, and column types, such as "*In the 'cinema' table 'cinema' based on the 'filmdom' database, what alternative name could be used for a column with the data type 'Text' that conveys a similar meaning to 'Movie'? Please return only one English word rather than a sentence.*" It was empirically demonstrated that this approach yielded superior results. Nevertheless, this method still has several limitations. For instance, in most cases, a table named "*happy_hour*" may have a column named "*HH_ID*", and the model is unaware that "*HH*" represents "*happy_hour*". To address these limitations, we made manual modifications.

2.3 Manual Correction

The output of LLM is characterized by instability. To ensure the efficacy of the dataset, it is necessary for us to undertake manual corrections on the entire dataset. In particular, as mentioned in Section 2.2, ChatGPT often fails to meet the robustness requirements when performing schema synonym substitution. Hence, we conducted a comprehensive and detailed manual modification of the entire nvBench-Rob dataset. This step constitutes the most critical and valuable aspect of dataset construction.

2.4 Dataset Analysis

We randomly divided nvBench into 3 parts according to the ratio of 80/4.5/15.5 in ncNet (Luo et al., 2021b). As a result, we obtained a development set consisting of 1182 pairs of (NL, VIS), involving a total of 104 databases. We performed robustness modifications (i.e. *NLQ reconstruction* and *schema synonymous substitution*) to both the 1182 pairs of (NL, VIS) and schemas in 104 databases. Eventually, three different levels of robustness datasets were obtained: **nvBench-Rob**$_{nlq}$, **nvBench-Rob**$_{schema}$, and **nvBench-Rob**$_{(nlq,schema)}$, corresponding to evaluating robustness modifications only on NLQs, only on table schemas, and on both NLQs and table schemas, respectively. The distribution of visualization chart types and the difficulty level of the DVQs are shown in Figure 2.

3 Robustness Analysis of Existing Text-to-Vis Models

As shown in Figure 3, the accuracy of existing text-to-vis models significantly decreased on the nvBench-Rob test set compared to the nvBench test set. Specifically, on a no-cross-domain split, the previous SOTA text-to-vis model, RGVisNet, achieved an accuracy of 85.17% on the nvBench test set, and other text-to-vis models also performed satisfactorily. However, even RGVisNet's accuracy dropped to 24.81% on the nvBench-Rob test set, which comprises both NLQs and data schema variants, marking a 60.36% decrease compared to its performance on the nvBench test set. This highlights the lack of robustness of the nvBench dataset and the high sensitivity of models trained on it to perturbations in model input.

For example, in the nvBench training set, data schemas like column names are explicitly mentioned in the NLQs, such as "ACC_Percent," enabling text-to-vis models to easily learn the explicit connection between NLQ and data schema. In the nvBench-Rob test set, NLQs no longer explicitly mention database column names, and sentences are reconstructed. Moreover, the column names in the

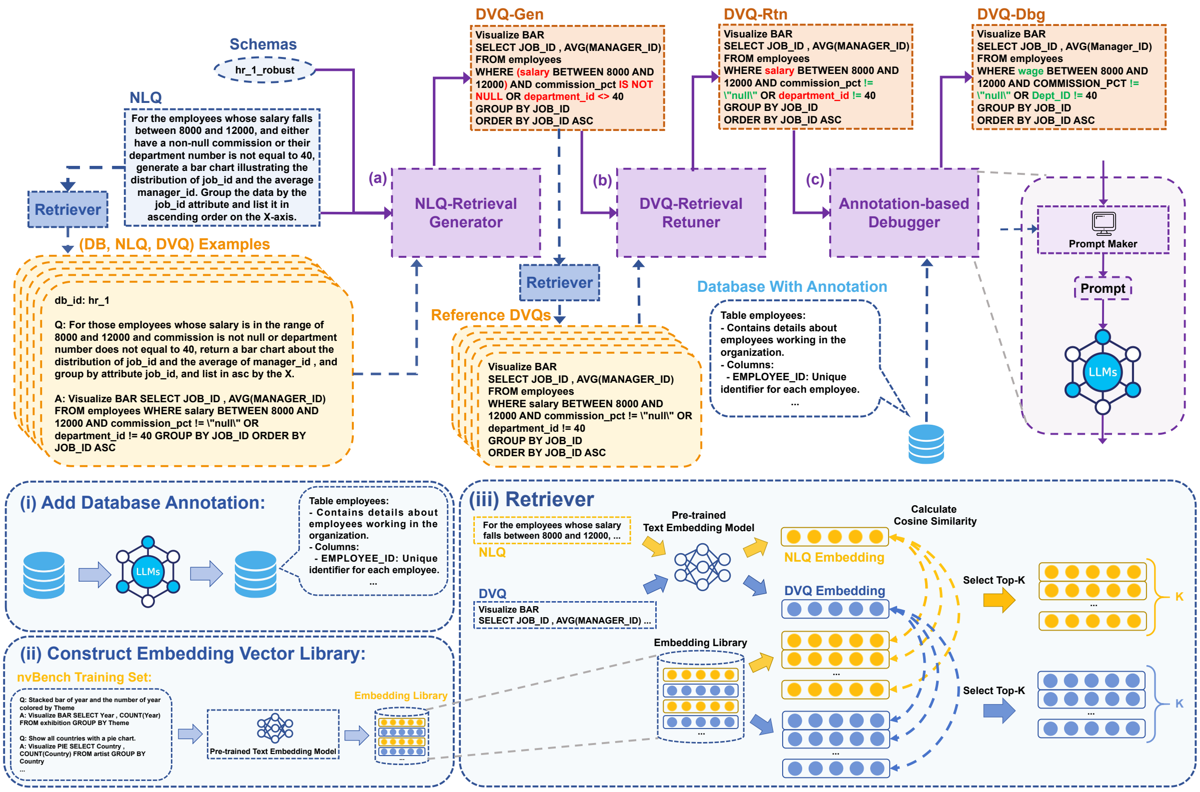


Figure 4: The working pipeline of our proposed GRED method, which includes three steps: (a) Input the NLQ into the Retriever to obtain the top-K (DB, NLQ, Schemas) instances, then input these instances along with the NLQ and Schemas into the ***NLQ-Retrieval Generator*** to get *DVQ_Rtn*; (b) Input the DVQ_Rtn into the Retriever to obtain the top-K DVQs, referred to as Reference DVQs, then input Reference DVQs along with DVQ_Rtn into the ***DVQ-Retrieval Retuner*** to get *DVQ_Rtn*; (c) Input the DVQ_Rtn and the annotated databases corresponding to Schemas into the ***Annotation-based Debugger*** to obtain the final result *DVQ_Dbg*.

database have been replaced with synonyms, for example, "ACC_Percent" has been replaced with "percentage_of_ACC." In these cases, previous text-to-vis models all fail to perform schema linking correctly, with RGVisNet still choosing the same column name "ACC_Percent" as in the training data, while models like Seq2Vis and Transformer are unable to generate the correct DVQ keywords. For more examples, please refer to Appendix C.

4 GRED: A Robustness Framework based on Retrieval-Augmented Generation

To enhance the robustness of text-to-vis models, we propose a novel RAG-based framework, named **GRED**. This framework comprises three core components: NLQ-Retrieval Generator, DVQ-Retrieval Retuner, and Annotation-based Debugger, aimed at addressing variants of NLQ, differences in programming styles, and changes in data schema, respectively. Before all the main processes of GRED, there are some preparatory works that need to be completed.

4.1 Preparatory Phase

The preparatory phase comprises two key steps: the establishment of an embedding vector library and the construction of an annotated database collection. Specifically, for the training set partitioned by nvBench, each NLQ and its corresponding DVQ are input into a pre-trained text embedding model to derive the associated embedding vectors, thereby populating the embedding vector library. The pre-trained text embedding model utilized in this work is the *text-embedding-3-large* model released by OpenAI. Regarding the construction of the annotated database collection, this process entails supplying database information to *GPT-3.5-Turbo* as prompts to generate corresponding NL annotations, which are then stored collectively.

4.2 Pipeline of GRED

NLQ-Retrieval Generator For the NLQs input into the text-to-vis system, GRED first converts them into embedding vectors using the text-embedding-3-large model mentioned in Section 4.1, and then calculates their cosine similarity with the embedding vectors of natural language questions in the embedding vector library constructed during the preparation. After that, it selects the top-K most similar natural language questions and assembles their corresponding examples into a generation prompt in ascending order of similarity. This prompt is then input into LLM like GPT-3.5-Turbo to generate the corresponding DVQ, referred to as DVQ_{gen}. It is worth mentioning that sorting in ascending order of similarity means placing examples with high similarity near the asking part of the prompt. For more details, please refer to Appendix D.2. The benefit of this approach is that it allows the LLM to achieve more accurate results based on the examples, thus reducing the model's hallucinations.

DVQ-Retrieval Retuner Similar to the retrieval process with NLQ, convert DVQ_{gen} into embedding vectors, and calculate the cosine similarity with the DVQ embedding vectors in the embedding library constructed in Section 4.1. Select the top-K most similar DVQs to construct retuning prompts, and then input them into LLM, such as GPT-3.5-Turbo, to mimic similar programming styles, thereby generating DVQ_{rtn}. The purpose of this step is to perform fine adjustments to the DVQ, such as choosing between "*IS NOT NULL*" and "*!= "null"*".

Annotation-based Debugger The examples in the embedding vector library constructed in Section 4.1 all come from nvBench, which means these examples do not contain data schema variations. This will cause LLMs to experience illusions when encountering data schema variants, resulting in the generation of DVQs with incorrect column names. To tackle this problem, an annotation-based debugger component is introduced. Specifically, this involves combining the database with NL annotations and DVQ_{rtn} into debugging prompts. Then, inputting them into GPT-3.5-Turbo and asking it to replace the inappropriate column names in DVQ_{rtn} to obtain the final DVQ_{dbg}.

In summary, the NLQ-Retrieval Generator ensures that the model's output is structurally similar to the target DVQ. The DVQ-Retrieval Retuner ensures that the model's output closely aligns with the target DVQ in terms of minor programming styles. Lastly, the Annotation-based Debugger guarantees the correctness of the data schema mentioned in the model's output DVQ.

5 Experiments and Analysis

In this section, we present the experimental setup and report the evaluation results. Through comparative analysis with other baselines, we demonstrate that our model outperforms baselines in terms of robustness, thus verifying the effectiveness of GRED. For case study, please refer to Appendix A.

5.1 Experimental Setup

Datasets. We evaluate the robustness of the previous text-to-vis model on the nvBench-Rob test set. The nvBench-Rob test set comprehensively evaluates the model's robustness from three different dimensions: the NLQ single-variant test set, the Data schema single-variant test set, and the dual-variant test set. Therefore, there are three sets of evaluations:

- **nvBench-Rob**$_{nlq}$: a testing set from nvBench-Rob, containing only NLQ variants, is specifically designed to test the robustness of models against NLQ variants.
- **nvBench-Rob**$_{schema}$: a testing set from nvBench-Rob, containing only data schema variants, is specifically designed to test the robustness of models against data schema variants.
- **nvBench-Rob**$_{(nlq,schema)}$: a testing set from nvBench-Rob, containing both NLQ variants and data schema variants, is specifically designed to test the robustness of models against both NLQ variants and data schema variants.

Baselines. We evaluate **GRED** and previous text-to-vis models on nvBench-Rob including **Seq2Vis** (Luo et al., 2021a), **Transformer** (Vaswani et al., 2017), and **RGVisNet** (Song et al., 2022), which is the previous SOTA model in text-to-vis. We conduct a detailed analysis of the robustness using their performance on nvBench-Rob.

Measurements. Following (Song et al., 2022; Luo et al., 2021a), four popular metrics, namely *Vis Accuracy*, *Data Accuracy*, *Axis Accuracy*, and *Overall Accuracy*, are used in our experiment to evaluate the performance.

Implementation Details. For the data preparation phase, specifically for generating NL annotations for each database, the parameters of the `openai.ChatCompletion.create` method are set as follows:

```
temperature=0.0,
frequency_penalty=0.0,
presence_penalty=0.0
```

Model	nvBench-Rob$_{nlq}$ Vis Acc.	Data Acc.	Axis Acc.	Acc.
Seq2Vis	93.91%	38.83%	42.23%	34.52%
Transformer	91.62%	48.22%	49.24%	36.04%
RGVisNet	96.37%	53.04%	70.12%	45.87%
GRED (Ours)	**97.63%**	**61.93%**	**88.41%**	**59.98%**

Table 1: Results in **nvBench-Rob**$_{nlq}$

Model	nvBench-Rob$_{schema}$ Vis Acc.	Data Acc.	Axis Acc.	Acc.
Seq2Vis	96.79%	18.02%	15.40%	14.55%
Transformer	92.22%	41.88%	38.16%	29.61%
RGVisNet	**98.33%**	55.09%	60.83%	44.91%
GRED	97.72%	**65.48%**	**85.03%**	**61.93%**

Table 2: Results in **nvBench-Rob**$_{schema}$

Model	nvBench-Rob$_{(nlq,schema)}$ Vis Acc.	Data Acc.	Axis Acc.	Acc.
Seq2Vis	94.16%	7.45%	7.11%	5.50%
Transformer	92.13%	22.59%	18.87%	12.77%
RGVisNet	96.76%	47.04%	34.07%	24.81%
GRED	**98.14%**	**58.48%**	**81.52%**	**54.85%**

Table 3: Results in **nvBench-Rob**$_{(nlq,schema)}$

However, during the formal working phase of GRED, the parameters of this function are set as follows:

```
temperature=0.0,
frequency_penalty=-0.5,
presence_penalty=-0.5
```

In addition, the large language model used in the experimental process is *GPT-3.5-Turbo* and uses the version released by OpenAI on *January 25, 2024*. The hyperparameter K, which means the retrieval number of NLQ and DVQ, in the experiment is *10*.

5.2 Experiment Result

As shown in Figure 3, previous text-to-vis models have achieved satisfactory performance on the nvBench test set. Even the simplest model, Seq2Vis, can easily achieve high precision. However, when the model input is perturbed, even the state-of-the-art(SOTA) model RGVisNet experiences a significant drop in accuracy.

In order to comprehensively assess the robustness of the models, we tested the models trained on nvBench with three test sets from nvBench-Rob. The results are presented in Table 1, Table 2 and Table 3. The SOTA text-to-vis model, RGVisNet, experienced a significant decline of 39.3% (85.17% vs. 45.87%) or 40.26% (85.17% vs. 44.91%) in accuracy on test sets with a single variation. The most notable difference was observed in nvBench-Rob, where the accuracy dropped by 60% (85.17% vs. 24.81%) compared to the original nvBench test set. Meanwhile, GRED demonstrated impressively high accuracy across the three test sets of nvBench-Rob, with an improvement of 16% (61.68% vs. 45.87%) on nvBench-Rob$_{nlq}$, 18.5% (63.45% vs. 44.91%) on nvBench-Rob$_{schema}$, and 32% (57.19% vs. 24.81%) on the most challenging nvBench-Rob$_{(nlq,schema)}$ test set. Such results indicate that GRED has a strong ability to resist interference with model inputs and demonstrate its excellent robustness.

5.3 Ablation Study

In this section, we conduct ablation studies to demonstrate the effectiveness and contribution of each design component in GRED. Specifically, we first evaluate GRED with all components included. Then, we remove some components of GRED to assess its performance with the following configurations: (i) utilizing only NLQ-Retrieval Generator without DVQ-Retrieval Retuner and Annotation-based Debugger **(w/o RTN&DBG)**; (ii) removing our Annotation-based Debugger **(w/o DBG)**; (iii) removing the DVQ-Retrieval Retuner **(w/o RTN)**.

The ablation study results shown in Table 4 confirm the importance of the three components designed in our proposed model. We observed that the NLQ-Retrieval Generator plays a crucial role in countering input perturbations caused by natural language variants, while the Annotation-based Debugger plays a key role in countering input perturbations caused by data schema variations. This is because they significantly improve the model's performance in their respective variant-specific test sets. The DVQ-Retrieval Retuner is also found to be very important since it helps LLM adjust the generated DVQ style to better match the dataset's style, thereby reducing errors in programming style and achieving higher accuracy. Therefore, these three components all contribute to the model's robustness.

6 Related Work

Text-to-Vis. Recent years, there has been significant growth in the adoption of Data Visualization (DV) in the fields of natural language processing (Ge et al., 2024; Dibia and Demiralp, 2019; Cui et al., 2019; Dibia and Demiralp, 2019), data min-

Model	nvBench-Rob$_{nlq}$	nvBench-Rob$_{schema}$	nvBench-Rob$_{(nlq,schema)}$
RGVisNet (SOTA)	45.87%	44.91%	24.81%
GRED (Ours)	59.98%	61.93%	**54.85%**
- w/o RTN&DBG	**62.77%**	42.13%	36.46%
- w/o RTN	61.08%	**62.10%**	51.90%
- w/o DBQ	61.68%	42.47%	38.57%

Table 4: Ablation Study Result on nvBench-Rob.

ing (Song et al., 2022; Qian et al., 2021; Ho et al., 2002; Fayyad et al., 2002), and database community (Tang et al., 2022; Hanrahan, 2006; Luo et al., 2021a; Vartak et al., 2017; Luo et al., 2018). To enhance the accessibility of DV, a task named text-to-vis has been proposed, which offers a mechanism to automatically transform natural language questions (NLQs) into DV charts. For instance, Luo et al. (2021a) delineated a methodology for synthesizing the NLQ-DV dataset, known as nvBench, predicated upon the renowned NL2SQL benchmark, Spider (Yu et al., 2018). A Seq2Seq model was subsequently trained on this benchmark, corroborating the viability of engendering DV queries from NLQs. RGVisNet (Song et al., 2022) represents another seminal study in which a DNN-based approach is employed to transform NLQ into DV.

Despite the abundance of text-to-vis models, the robustness of these models remains underexplored. We not only proposed the first comprehensive robustness evaluation dataset for text-to-vis tasks but also introduced a framework based on RAG with LLMs to address perturbations in the model's input. With the benchmark and the method proposed in this paper, nvBench-Rob would become a popular dataset for evaluating the robustness of text-to-vis models and inspire further research in the *NLP for Data Visualization* direction.

Robustness in NLP. The robustness of a model is a crucial evaluation criterion for its deployment in real-life scenarios. In the field of NLP, there have been numerous studies on model robustness. Some studies have investigated the influence of model inputs on robustness (Hendrycks et al., 2019, 2020; Chen et al., 2022; Yan et al., 2022). Besides, some studies have introduced evaluation metrics to evaluate model robustness across various domains(Wang et al., 2023; Zhao et al., 2023). A comprehensive survey on Robustness in NLP can be found in (Wang et al., 2022).

We are the first to explore the robustness of the text-to-vis task and designed the first text-to-vis robustness evaluation dataset, nvBench-Rob, which expands the frontiers of existing robustness research.

RAG in NLP Retrieval-Augmented Generation (RAG) technology has become the primary method to fully utilize the capabilities of LLMs in downstream tasks (Kim et al., 2023; Ma et al., 2023; Long et al., 2023; Pozzobon et al., 2023; Yu et al., 2023; Shao et al., 2023; Mavi et al., 2023). It has achieved notable results in various tasks such as open-domain QA (Izacard and Grave, 2021; Trivedi et al., 2023; Li et al., 2023), dialogue (Cai et al., 2019a,b; Peng et al., 2023), domain-specific question answering (Cui et al., 2023) and code generation (Zhou et al., 2023).

We introduced a RAG-based framework called GRED, which effectively addresses this issue by breaking down the visualization generation process into subprocess, progressively approximating the ultimate goal. In summary, we are the first to validate the effectiveness of the RAG technique in the robust text-to-vis scenario.

7 Conclusion

Robustness is a crucial factor for evaluating model performance. In this study, we introduce the first comprehensive robustness benchmark, nvBench-Rob, for evaluating the robustness of text-to-vis models. Then, we found that the performance of existing text-to-vis models is not satisfactory on the robustness scenario. Finally, we propose a novel framework named GRED based on the RAG-techniques using LLMs, which addresses challenges posed by NLQ variations, programming style differences, and data schema variations through three components: NLQ-Retrieval Generator, DVQ-Retrieval Retuner, and Annotation-based Debugger. Our experiments reveal the inherent difficulty of developing robust text-to-vis models, and simultaneously demonstrate the effectiveness of GRED through extensive empirical validation.

Limitations

In this work, we are the first to comprehensively study the robustness of the text-to-vis task. We proposed the first comprehensive robustness evaluation set for text-to-vis and developed a framework based on RAG technology for LLMs to tackle the issue of insufficient robustness of text-to-vis models. However, this work only focuses on the single-turn text-to-vis task and does not address the more complex schema linking challenges associated with multi-turn interactions. We believe that researching the robustness of multi-turn text-to-vis tasks is a promising direction for future work.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2019a. Skeleton-to-response: Dialogue generation guided by retrieval memory. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1219–1228, Minneapolis, Minnesota. Association for Computational Linguistics.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019b. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1866–1875, Hong Kong, China. Association for Computational Linguistics.

Howard Chen, Jacqueline He, Karthik Narasimhan, and Danqi Chen. 2022. Can rationalization improve robustness? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3792–3805, Seattle, United States. Association for Computational Linguistics.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *Preprint*, arXiv:2306.16092.

Weiwei Cui, Xiaoyu Zhang, Yun Wang, He Huang, Bei Chen, Lei Fang, Haidong Zhang, Jian-Guan Lou, and Dongmei Zhang. 2019. Text-to-viz: Automatic generation of infographics from proportion-related natural language statements. *IEEE transactions on visualization and computer graphics*, 26(1):906–916.

Victor Dibia and Çağatay Demiralp. 2019. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications*, 39(5):33–46.

Longxu Dou, Yan Gao, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, and Jian-Guang Lou. 2023. Multispider: towards benchmarking multilingual text-to-sql semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12745–12753.

Usama M Fayyad, Georges G Grinstein, and Andreas Wierse. 2002. *Information visualization in data mining and knowledge discovery*. Morgan Kaufmann.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*.

Yan Ge, Junqiu Wei, Yuanfeng Song, Chen Zhang, and Raymond Chi-Wing Wong. 2024. Automatic data visualization generation from chinese natural language questions. In *The 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

Virgilio Gómez-Rubio. 2017. ggplot2 - elegant graphics for data analysis (2nd edition). *Journal of Statistical Software*, 77:1–3.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Pat Hanrahan. 2006. Vizql: a language for query, analysis and visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 721–721.

D. Hendrycks, K. Lee, and M. Mazeika. 2019. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning (ICML)*.

D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Association for Computational Linguistics (ACL)*.

TuBao Ho, TrongDung Nguyen, and DungDuc Nguyen. 2002. Visualization support for a user-centered kdd process. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 519–524.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880. Association for Computational Linguistics.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. 2023. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 996–1009, Singapore. Association for Computational Linguistics.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1774–1793, Toronto, Canada. Association for Computational Linguistics.

Deqing Li, Honghui Mei, Yi Shen, Shuang Su, Wenli Zhang, Junting Wang, Ming Zu, and Wei Chen. 2018. Echarts: A declarative framework for rapid construction of web-based visualization. *Vis. Informatics*, 2:136–146.

Quanyu Long, Wenya Wang, and Sinno Pan. 2023. Adapt in contexts: Retrieval-augmented domain adaptation via in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6525–6542, Singapore. Association for Computational Linguistics.

Yuyu Luo, Xuedi Qin, Nan Tang, Guoliang Li, and Xinran Wang. 2018. Deepeye: Creating good data visualizations by keyword search. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1733–1736.

Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021a. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 1235–1247, New York, NY, USA. Association for Computing Machinery.

Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2021b. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

Vaibhav Mavi, Abulhair Saparov, and Chen Zhao. 2023. Retrieval-augmented chain-of-thought in semi-structured domains. In *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 178–191, Singapore. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *Preprint*, arXiv:2302.12813.

Luiza Pozzobon, Beyza Ermis, Patrick Lewis, and Sara Hooker. 2023. Goodtriever: Adaptive toxicity mitigation with retrieval-augmented models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5108–5125, Singapore. Association for Computational Linguistics.

Xin Qian, Ryan A Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Joel Chan. 2021. Learning to recommend visualizations from data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1359–1369.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2018. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23:341–350.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya G. Parameswaran. 2016. Effortless data exploration with zenvisage: An expressive and interactive visual analytics system. *Proc. VLDB Endow.*, 10:457–468.

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*, pages 1646–1655.

Jiawei Tang, Yuyu Luo, Mourad Ouzzani, Guoliang Li, and Hongyang Chen. 2022. Sevi: Speech-to-visualization through neural machine translation. In *Proceedings of the 2022 International Conference on Management of Data*, pages 2353–2356.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. 2017. Towards visualization recommendation systems. *Acm Sigmod Record*, 45(4):34–39.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Shiqi Wang, Zheng Li, Haifeng Qian, Chenghao Yang, Zijian Wang, Mingyue Shang, Varun Kumar, Samson Tan, Baishakhi Ray, Parminder Bhatia, Ramesh Nallapati, Murali Krishna Ramanathan, Dan Roth, and Bing Xiang. 2023. ReCode: Robustness evaluation of code generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13818–13843, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Haohan Wang, and Diyi Yang. 2022. Measure and improve robustness in NLP models: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4569–4586, Seattle, United States. Association for Computational Linguistics.

Jun Yan, Yang Xiao, Sagnik Mukherjee, Bill Yuchen Lin, Robin Jia, and Xiang Ren. 2022. On the robustness of reading comprehension models to entity renaming. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 508–520, Seattle, United States. Association for Computational Linguistics.

Guoxin Yu, Lemao Liu, Haiyun Jiang, Shuming Shi, and Xiang Ao. 2023. Retrieval-augmented few-shot text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6721–6735, Singapore. Association for Computational Linguistics.

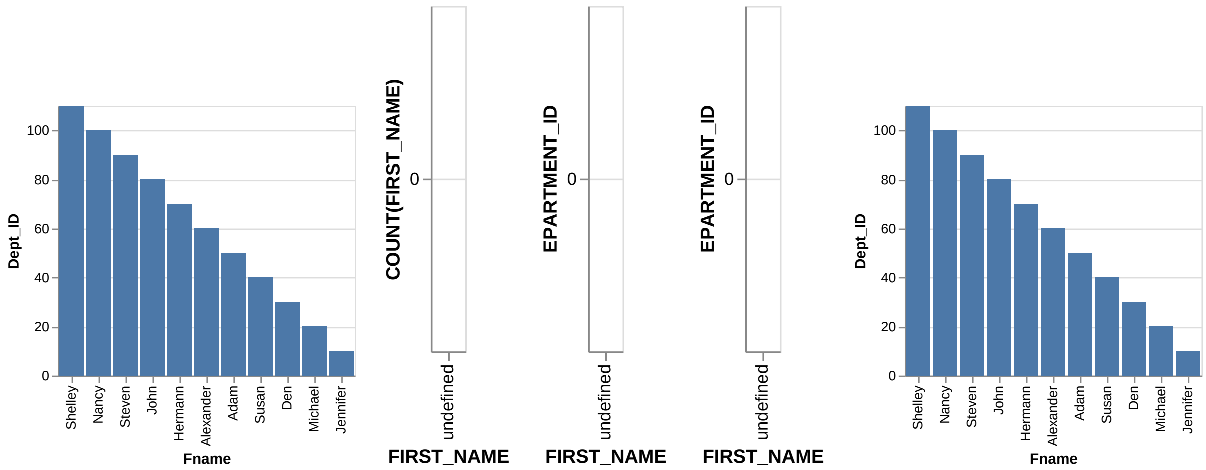
Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. 2023. RobuT: A systematic study of table QA robustness against human-annotated adversarial perturbations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6081, Toronto, Canada. Association for Computational Linguistics.

Shuyan Zhou, Uri Alon, Frank F. Xu, Zhengbao Jiang, and Graham Neubig. 2023. Docprompting: Generating code by retrieving the docs. In *The Eleventh International Conference on Learning Representations*.

A Case Study

NLQ	Present the ***department_id*** by ***first name*** in a histogram, with the Y-axis organized in descending order, please.
Target DVQ	Visualize BAR SELECT ***Fname***, ***Dept_ID*** FROM employees ORDER BY ***Dept_ID*** DESC → *Figure 5a*
Seq2Vis	Visualize BAR SELECT FIRST_NAME , COUNT (FIRST_NAME) FROM dogs ORDER BY COUNT (LAST_NAME) DESC → *Figure 5b*
Transformer	Visualize BAR SELECT FIRST_NAME , DEPARTMENT_ID FROM employees ORDER BY DEPARTMENT_ID DESC → *Figure 5c*
RGVisNet	Visualize BAR SELECT FIRST_NAME , DEPARTMENT_ID FROM employees ORDER BY DEPARTMENT_ID DESC → *Figure 5d*
GRED	Visualize BAR SELECT **Fname** , **Dept_ID** FROM employees ORDER BY **Dept_ID** DESC → *Figure 5e*



(a) Target (b) Seq2Vis ✘ (c) Transformer ✘ (d) RGVisNet ✘ (e) GRED ✔

Table 5: Case Study. DVQs generated by other baselines like RGVisNet and GRED, together with their corresponding visualization charts (Errors are marked with red colors).

Table 5 presents a case study illustrating the DVQ generated by GRED and previous SOTA model RGVisNet. The charts generated by these models are also shown in Table 5. As illustrated by Table 5, Seq2Vis generate incorrect column names and aggregation keywords on y-axis, resulting in no chart being shown in Table 5b. RGVisNet and Transformer generate DVQs with the correct aggregation keywords. However, due to the lack of robustness to perturbations in model inputs, both RGVisNet and Transformer fail to accurately generate the column names "Fname" and "Dept_ID". Instead, they retain the column names "FIRST_NAME" and "DEPARTMENT_ID" from the training set, which also results in no chart being produced in Table 5d and Table 5c. Unlike the aforementioned models, GRED is capable of not only generating a structure that is identical to the target query but also producing the correct column names, thereby resulting in the accurate charts as shown in Table 5e.

B Detailed Definitions of the Evaluation Metrics

- Overall Accuracy: This metric measures exact matches between the predicted DV query and the target DV query. The accuracy calculation formula is:

  $$\texttt{Acc.} = N_c \,/\, N$$

  where N_c represents the number of the matched DV queries and N represents the size of the test set. This metric directly reflects the comprehensive performance of the model.

- Vis Accuracy: Each DVQ consists of three types of components: the DV chart type, the x/y-axis, and the data transformation. This evaluation metric reflects the matches between the generated DVQ and the target DVQ in terms of the type of DV chart. The accuracy calculation formula is:

  $$\texttt{Vis Acc.} = N_{Vis} \,/\, N$$

  Where N_{Vis} represents the number of DV chart types in the generated DVQs that match the DV chart types in the target DVQs.

- Axis Accuracy: This evaluation metric calculates the matches of the x/y axis components between the generated DVQs and the real DVQs. The accuracy calculation formula is:

  $$\texttt{Axis Acc.} = N_{Axis} \,/\, N$$

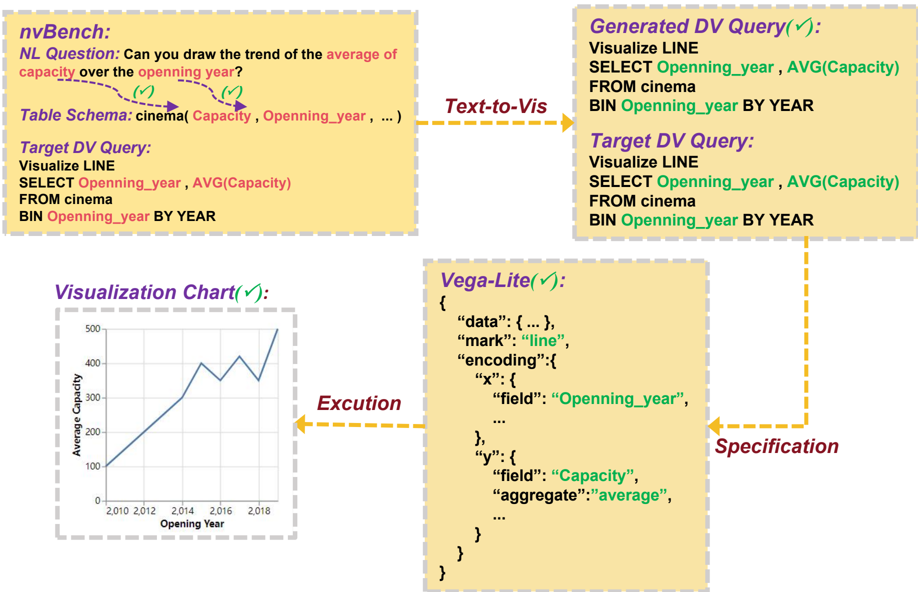
  where N_{Axis} represents the number of x/y-axis components in the generated DVQs that match the x/y-axis components in the target DVQs.

- Data Accuracy: Similarly, this measurement reflects the matches of the data transformation components between the generated DVQs and the target DVQs. The accuracy calculation formula is:

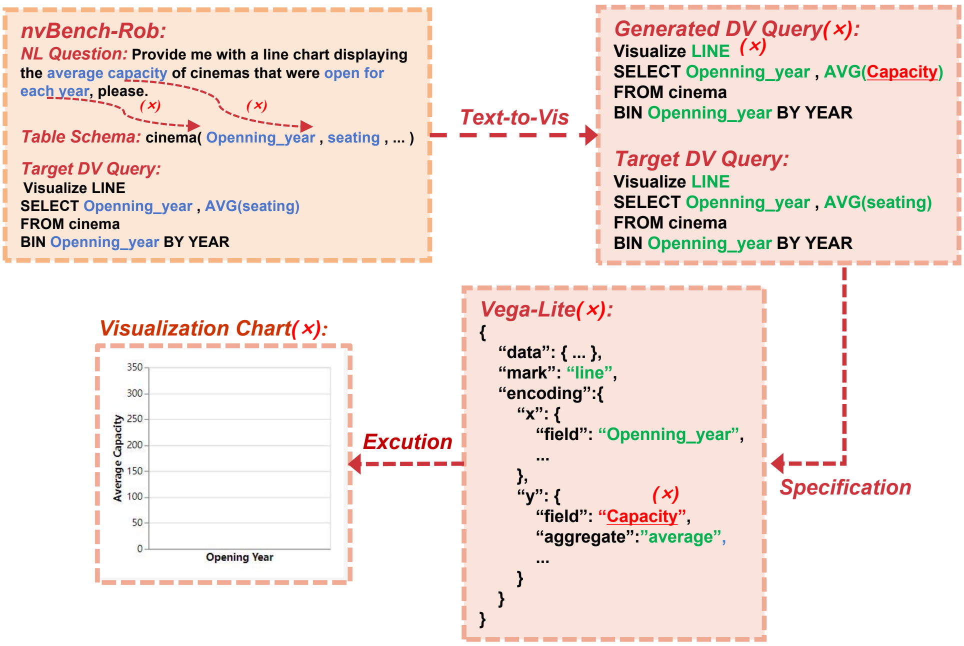
  $$\texttt{Data Acc.} = N_{Data} \,/\, N$$

  where N_{Data} represents the number of data transformation components in the generated DVQs that match the Ddata transformation components in the target DVQs.

C Robustness Analysis Cases



(a) Correct Case in the original Text-to-Vis testing set



(b) Failure Case of existing Text-to-Vis Models on the Robustness scenario.

Figure 5: Robustness Analysis Cases

Figures 5a and Figures 5b show examples of previous text-to-vis models successfully generating accurate data visualizations on the original nvBench test set, as well as instances where they fail to produce the final data visualizations due to the addition of NLQ variants and data schema variants. It is not difficult to observe that when the explicit alignment between NLQ and data schema is eliminated, previous text-to-vis models are unable to correctly perform schema linking, even when the data schema has the same meaning as the data schemas in the original training set.

D Prompts

D.1 Database Annotation Generator

Role: SYSTEM
Content:
You are a data mining engineer with ten years of experience in data visualization.

Role: USER
Content:
Please generate detailed natural language annotations to the following database schemas.
Database Schemas:
Table departments, columns = [* , Dept_ID , Dept_NAME , Manager_ID , Location_ID]
Table job_history , columns = [* , employee_id , START_DATE , END_DATE , JOB_ID , Dept_ID]
Table jobs , columns = [* , JOB_ID , JOB_TITLE , minimum_salary , maximum_salary]
Foreign_keys = [job_history.JOB_ID = jobs.JOB_ID , job_history.Dept_ID = departments.Dept_ID]

Natural Language Annotations:
A:
Table departments:
- Stores data related to different departments within an organization.
- Columns:
- Dept_ID: Unique identifier for each department.
- Dept_NAME: Name of the department.
- Manager_ID: Identifier of the manager of the department.
- Location_ID: Identifier of the location where the department is situated.

Table job_history:
- Stores historical data of job changes for employees.
- Columns:
- employee_id: Identifier of the employee.
- START_DATE: Start date of the job role. - END_DATE: End date of the job role.
- JOB_ID: Identifier of the job role.
- Dept_ID: Identifier of the department during the job role.

Table jobs:
- Contains information about different job roles.
- Columns:
- JOB_ID: Unique identifier for each job role.
- JOB_TITLE: Title of the job role.
- minimum_salary: Minimum salary for the job role.
- maximum_salary: Maximum salary for the job role.

Foreign Keys:
- job_history.JOB_ID references jobs.JOB_ID, linking job history to specific job roles.
- job_history.Dept_ID references departments.Dept_ID, connecting job history to departments.

Database Schemas:
*****[new Database Schemas]*****

Natural Language Annotations:
A:

D.2 NLQ-Retrieval Generator Prompt

Role: SYSTEM
Content:
Please follow the syntax in the examples instead of SQL syntax.

Role: USER
Content:
Given Natural Language Questions, Generate DVQs based on their correspoding Database Schemas.

*****[Top-$K-1$ Examples]*****

Database Schemas:
Table Has_Pet, columns = [* , StuID , PetID]
Table Pets, columns = [* , PetID , PetType , pet_age , weight]
Table Student, columns = [* , StuID , LName , Fname , Age , Sex , Major , Advisor , city_code]
Foreign_keys = [Has_Pet.StuID = Student.StuID , Has_Pet.PetID = Pets.PetID]
#
Chart Type: [BAR , PIE , LINE , SCATTER]
Natural Language Question:
"Find the id and weight of all pets whose age is older than 1 Visualize by bar chart, sort by the Y-axis from high to low."
Data Visualization Query:
A: Visualize BAR SELECT PetID , weight FROM pets WHERE pet_age > 1 ORDER BY weight DESC

Database Schemas:
Table Has_Pet, columns = [* , StuID , PetID]
Table Pets, columns = [* , PetID , PetType , pet_age , weight]
Table Student, columns = [* , StuID , LName , Fname , Age , Sex , Major , Advisor , city_code]
Foreign_keys = [Has_Pet.StuID = Student.StuID , Has_Pet.PetID = Pets.PetID]
#
Chart Type: [BAR , PIE , LINE , SCATTER]
Natural Language Question:
"Find the id and weight of all pets whose age is older than 1 Visualize by bar chart, sort in descending by the names."
Data Visualization Query:

D.3 DVQ-Retrieval Retuner

Role: SYSTEM
Content:
The Reference Data Visualization Queries(DVQs) all comply with the syntax of DVQ. Please follow the syntax of the referenced DVQ to modify the Original DVQ.
Role: USER
Content:
Reference DVQs:
*****[Top-$K-1$ DVQs]*****
10 - Visualize BAR SELECT JOB_ID , SUM(DEPARTMENT_ID) FROM employees WHERE first_name LIKE '%D%' OR first_name LIKE '%S%' GROUP BY JOB_ID ORDER BY SUM(DEPARTMEN)

Given the Reference DVQs, please modify the Original DVQ to mimic the style of the Reference DVQs.
NOTE: Do not Modify the column name in Original DVQ. Especially do not Modify the column names in the ORDER clause!
Original DVQ:
Visualize BAR SELECT JOB_ID , COUNT(DISTINCT JOB_ID) FROM employees WHERE DEPARTMENT_ID = (SELECT DEPARTMENT_ID FROM departments WHERE DEPARTMENT_NAME = Finance)
A: Let's think step by step!

Role: ASSISTANT
Content:
Modified DVQ:
Visualize BAR SELECT JOB_ID , COUNT(JOB_ID) FROM employees AS T1 JOIN departments AS T2 ON T1.DEPARTMENT_ID = T2.DEPARTMENT_ID WHERE T2.DEPARTMENT_NAME = 'Finance' GROUP BY JOB_ID

Role: USER
Content:
Reference DVQs:
*****[Top-K DVQs]*****

Given the Reference DVQs, please modify the Original DVQ to mimic the style of the Reference DVQs.
NOTE: Do not Modify the column name in Original DVQ. Especially do not Modify the column names in the ORDER clause!
Original DVQ:
[Original DVQ]
A: Let's think step by step!

D.4 Annotation-based Debugger

Role: SYSTEM
Content:
NOTE: Don't replace column names in Original DVQ that already exist in the database schemas, especially column names in GROUP BY Clause!

Role: USER
Content:
Please generate detailed natural language annotations to the following database schemas.
Database Schemas:
Table departments, columns = [* , Dept_ID , Dept_NAME , Manager_ID , Location_ID]
Table job_history , columns = [* , employee_id , START_DATE , END_DATE , JOB_ID , Dept_ID]
Table jobs , columns = [* , JOB_ID , JOB_TITLE , minimum_salary , maximum_salary]
Foreign_keys = [job_history.JOB_ID = jobs.JOB_ID , job_history.Dept_ID = departments.Dept_ID]

Natural Language Annotations:
A:
Table departments:
- Stores data related to different departments within an organization.
- Columns:
- Dept_ID: Unique identifier for each department.
- Dept_NAME: Name of the department.
- Manager_ID: Identifier of the manager of the department.
- Location_ID: Identifier of the location where the department is situated.

Table job_history:
- Stores historical data of job changes for employees.
- Columns:
- employee_id: Identifier of the employee.
- START_DATE: Start date of the job role. - END_DATE: End date of the job role.
- JOB_ID: Identifier of the job role.
- Dept_ID: Identifier of the department during the job role.

Table jobs:
- Contains information about different job roles.
- Columns:
- JOB_ID: Unique identifier for each job role.
- JOB_TITLE: Title of the job role.
- minimum_salary: Minimum salary for the job role.
- maximum_salary: Maximum salary for the job role.

Foreign Keys:
- job_history.JOB_ID references jobs.JOB_ID, linking job history to specific job roles.
- job_history.Dept_ID references departments.Dept_ID, connecting job history to departments.

Given Database Schemas and their corresponding Natural Language Annotations, Please replace the column names in the Data Visualization Query(DVQ, a new Programming Language abstracted from Vega-Zero) that do not exist in the database.
NOTE: Don't replace column names in Original DVQ that already exist in the database schemas, especially column names in GROUP BY Clause!
Original DVQ:

Visualize BAR SELECT jobid , COUNT(jobid) FROM employees AS T1 JOIN departments AS T2 ON T1.DEPARTMENT_ID = T2.DEPARTMENT_ID WHERE T2.DEPARTMENT_NAME = 'Finance' GROUP BY FIRST_NAME
A: Let's think step by step!

Role: ASSISTANT
Content:
Revised DVQ:
Visualize BAR SELECT JOB_ID , COUNT(JOB_ID) FROM employees AS T1 JOIN departments AS T2 ON T1.Dept_ID = T2.Dept_ID WHERE T2.Dept_NAME = 'Finance' GROUP BY FIRST_NAME
Role: SYSTEM
Content:
NOTE: Don't replace column names in Original DVQ that already exist in the database schemas, especially column names in GROUP BY Clause!

Role: USER
Content:
Please generate detailed natural language annotations to the following database schemas.
Database Schemas:
******[Database Schemas]******

Natural Language Annotations:
******[NL Annotation of Database Schemas]******

Given Database Schemas and their corresponding Natural Language Annotations, Please replace the column names in the Data Visualization Query(DVQ, a new Programming Language abstracted from Vega-Zero) that do not exist in the database.
NOTE: Don't replace column names in Original DVQ that already exist in the database schemas, especially column names in GROUP BY Clause!
Original DVQ:
[Original DVQ]
A: Let's think step by step!