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

Jinwei Lu* Yuanfeng Song†, Haodi Zhang*§ Chen Zhang¶, Kaishun Wu§, Raymond Chi-Wing Wong‡
*Shenzhen University, Shenzhen, China †AI Group, WeBank Co., Ltd, Shenzhen, China
‡HKUST, Hong Kong, China §HKUST(GZ), Guangzhou, China ¶PolyU, Hong Kong, China

***Abstract*—Text-to-Vis is an emerging task in the data engineering and mining 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 Prompt4Vis in the Text-to-Vis field, GRED performs better in terms of model robustness, with a 40% increase in accuracy on the proposed nvBench-Rob dataset.**

***Index Terms*—text-to-visualization, model robustness, large language model, retrieval-augmented generation, input variation**

I. 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 [1], ggplot2 [2], ZQL [3], and ECharts [4]—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-vis* has been proposed, which offers a mechanism to automatically transform natural language questions (NLQs) into DV charts [5], [6]. Exemplified in Figure 1, the text-to-vis system requires users to upload a database and then simply ask a 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, which refers to its ability to maintain consistent performance and accurate predictions in the face of input variations, plays a crucial 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 IV), 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-to-vis dataset nvBench [5] 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) [7]–[11]. This framework comprises three core components: NLQ-Retrieval Generator, DVQ-Retrieval Retuner, and Annotation-based Debugger, aimed at

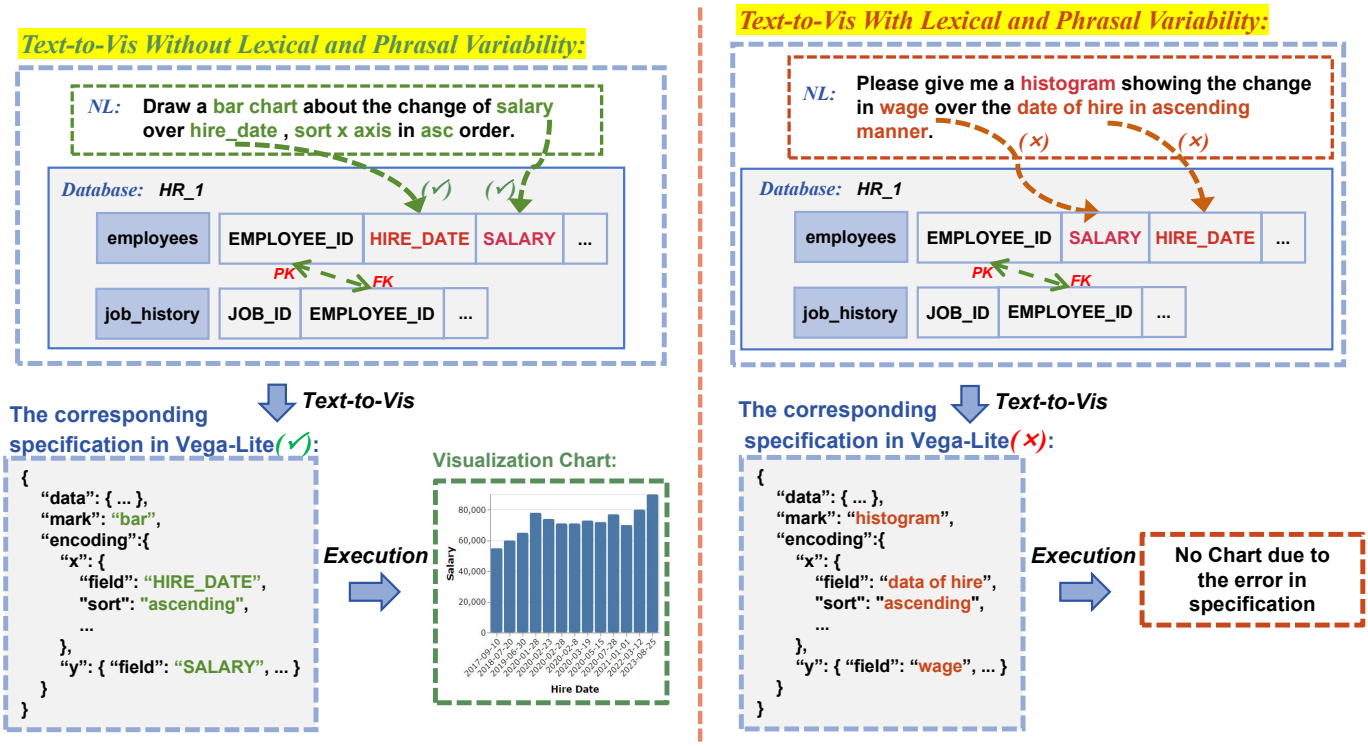


Fig. 1: (a) Text-to-vis is dedicated to converting natural language questions (NLQs) into data visualizations (DVs). The current approach heavily relies on explicit matching between words within the NLQs and the table schema. (b) The robustness of existing text-to-vis methods is limited. When small variations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

addressing variants of NLQs, differences in programming styles, and changes in data schema, respectively. [1]

Specifically, in the preparation phase, GRED utilizes a pre-trained text embedding model [12], [13] 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 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 almost 20% on the single-variant test set and 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 and release nvBench-Rob [2], 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

[1]DVQ refers to Data Visualization Query [5], [6], which is a widely-used intermediate representation that connects NLQ with the DVLs like Vega-Lite and ECharts.

[2]The robustness test set nvBench-Rob, the source code of the newly designed framework GRED, and the online technical report are all available at https://1drv.ms/f/c/80e862c5ba9a0a46/EmSwdjXJqR9EhCrns0cYzhYBLOe28qMF8ICZIHolpzqCAA?e=e90Kfs

the high sensitivity of text-to-vis models to input perturbations and inconsistencies in programming styles. It provides an innovative paradigm for leveraging LLMs to tackle robustness issues in the text-to-vis field.
- We conducted extensive experiments on the new robustness test set. Through a detailed analysis of the experimental results, we demonstrate that existing text-to-vis models, when trained on benchmark datasets, exhibit a lack of generalization to practical situations. In contrast, our proposed framework, GRED, outperforms current state-of-the-art (SOTA) models by 30% in accuracy, suggesting that GRED demonstrates superior robustness, enabling better generalization to real-world contexts.

The subsequent sections of this paper are organized as follows. Section II provides a review of the related literature. Section III outlines the construction process and features of the proposed dataset, nvBench-Rob. Section IV conducts a detailed analysis of the performance of current models on the new robustness test set, nvBench-Rob, through case studies. Section V introduces the GRED framework, which is founded on RAG and tailored to real-world scenarios, offering an elaborate explanation. Experimental findings are presented in Section VI. Lastly, Section VII serves as the conclusion of our paper.

II. Related Work

This research intersects three key fields: automatic data visualization, robustness in data engineering, and RAG for LLM. In this section, we will provide a concise overview of the most relevant works within each domain.

A. Automatic Data Visualization

Recent years, there has been significant growth in the adoption of Data Visualization (DV) in the fields of natural language processing [14], data mining [6], [15]–[17], and database community [5], [18]–[21]. Various advanced techniques have been developed to simplify the use of DV. We will cover two typical approaches: text-to-vis and DV Recommendation.

Text-to-vis primarily aims to convert a natural language question (NLQ) into its equivalent DV, simplifying the process for non-specialist users. The prevailing approach treats this conversion similarly to machine translation, engaging deep neural networks to map NLQs to DVs [22]–[24]. For instance, Cui *et al.* introduced the concept of text-to-viz, employing rule-based systems to convert text into infographics [25]. Luo *et al.* also delineated a methodology for synthesizing the NLQ-DV dataset, known as nvBench, predicated upon the renowned NL2SQL benchmark, Spider [26]. A Seq2Seq model was subsequently trained on this benchmark [5], corroborating the viability of engendering DV queries from NLQs. RGVisNet [6] represents another seminal study in which a DNN-based approach is employed to transform NLQ into DV. On the other hand, automated DV recommendation systems output probable DVs from datasets without any NLQ involvement. DataEye [21] simplifies this problem into recognition, ranking, and selection steps. Qian *et al.* [15] introduced an end-to-end learning-based approach for constructing DVs from extensive datasets.

Despite the abundance of models in text-to-vis, the robustness of these models remains underexplored. We introduce nvBench-Rob, the first dataset designed to comprehensively evaluate the robustness of existing text-to-vis models. Furthermore, we propose a novel RAG-based framework called GRED, designed to address perturbations in the model's input from three aspects: NLQ variants, programming style differences, and data schema variants. 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 Visualization* direction.

B. Robustness in Data Engineering

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 [27]–[30]. Other studies have proposed domain-specific optimization methods for model robustness [31]–[33]. Besides, some studies have introduced evaluation metrics to evaluate model robustness across various domains [33]. In another study, Zhao *et al.* [31] introduced a benchmark named ROBUT, which incorporates human-annotated perturbations in table headers, table content, and questions to evaluate the robustness of Table QA models. A comprehensive survey on Robustness in NLP can be found in [34].

In our research, to thoroughly assess the robustness of text-to-vis models, we introduce a robustness evaluation dataset called nvBench-Rob. Through collaboration between a LLM and human annotators, this dataset not only removes explicit mentions of column names in NLQs but also integrates a variety of language styles in the NLQs. Furthermore, it encompasses a variety of naming conventions for the table schemas in the dataset, thus creating an exceptionally robust evaluation dataset.

C. RAG for LLM

Retrieval-Augmented Generation (RAG) technology has become the primary method to fully utilize the capabilities of LLMs in downstream tasks [35], [36]. It has achieved notable results in various tasks such as open-domain QA [37], dialogue [38], domain-specific question answering [39] and code generation [40]. Additionally, a noteworthy work is PURPLE [41], which combines RAG technology, in-context learning techniques, and the use of small language models to assist LLM. This work has achieved new SOTA results on the NL2SQL benchmark dataset Spider [26] and has been published by ICDE'23.

We introduced a RAG-based framework called GRED, which effectively addresses this issue by breaking down the visualization query generation process into subprocess, progressively approximating the ultimate goal. This approach

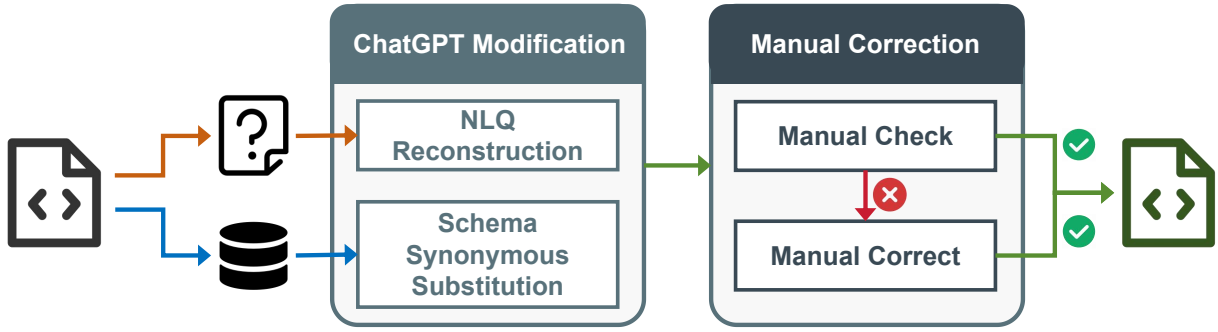


Fig. 2: The pipeline of dataset construction. In the ChatGPT Modification step, the NLQs from the development set separated from nvBench were restructured, and synonymous replacements were made to the table schemas of the databases involved; subsequently, in the Manual Correction step, the entire new dataset was manually inspected, and cases that were incorrect were manually corrected.

fully utilizes the role of outdated external knowledge. We are the first to validate the effectiveness of the RAG technique in the robust text-to-vis scenario.

III. Robustness Dataset: nvBench-Rob

In this chapter, we will provide a detailed introduction to the construction process of nvBench-Rob, which can be broadly divided into two steps: ChatGPT Modification (Section III-B) and Manual Correction (Section III-C). The ChatGPT Modification step can further be divided into two sub-processes: NLQ Reconstruction and Schema Synonymous Substitution.

A. Overview

The construction process of nvBench-Rob is shown in Figure 2. 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.

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.

B. 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 [11], 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 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 [42] 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.

C. 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 III-B, ChatGPT often fails to meet the

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

Fig. 3: Statistics of the nvBench-Rob Dataset

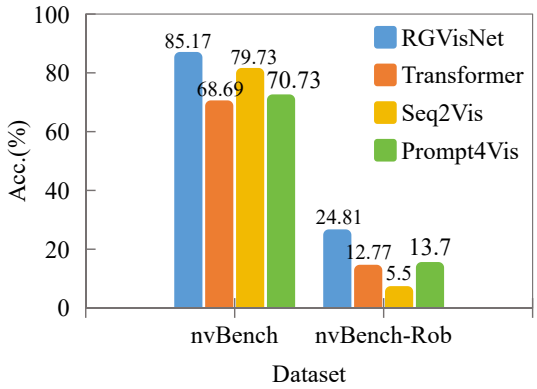


Fig. 4: The performance of existing text-to-vis models dramatically **drops** on the nvBench-Rob datasets.

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.

IV. Analysis on nvBench-Rob

In this chapter, we will first conduct a detailed analysis of nvBench-Rob and present its comprehensive features. Then, through the analysis of experimental results, we will examine the performance of existing text-to-vis models on this dataset, thereby reflecting their ability to generalize to real-world scenarios.

A. Dataset Analysis

Following the 80/4.5/15.5 ratio identified in ncNet [43], we randomly divided nvBench into three segments. Ultimately, we obtained a development set consisting of 1182 pairs of (NL, VIS), involving a total of 104 databases. We then constructed the following three levels of robustness test sets:

- **nvBench-Rob**$_{nlq}$: For the 1182 pairs of (NL, VIS) in the development set, we applied NLQ Reconstruction and Manual Correction, as described Section III, to create a test set that underwent only NLQ robustness modifications. This test set is used to evaluate how text-to-vis models perform when faced with NLQ variants, simulating questions posed by real users.
- **nvBench-Rob**$_{schema}$: For the 104 databases included in the development set, we conducted Schema Synonymous Substitution and Manual Correction, following the guidelines in Section III, to robustly modify the databases. The resulting test set mirrors the creation of databases under diverse naming conventions commonly seen in real-world scenarios, evaluating the ability of text-to-vis models to handle diverse schema naming rules.
- **nvBench-Rob**$_{(nlq,schema)}$: For the 1182 pairs of (NL, VIS) and the 104 databases in the development set, we carried out comprehensive robustness modifications included NLQ Reconstruction and Schema Synonymous Substitution. As a result, we created the most challenging and representative test set of real-world scenarios for text-to-vis models.

All three of these test sets have identical distributions of visualization chart types and DVQ difficulty levels, as illustrated in Figure 3.

B. Robustness Analysis on nvBench-Rob

As shown in Figure 4, 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 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.

V. 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.

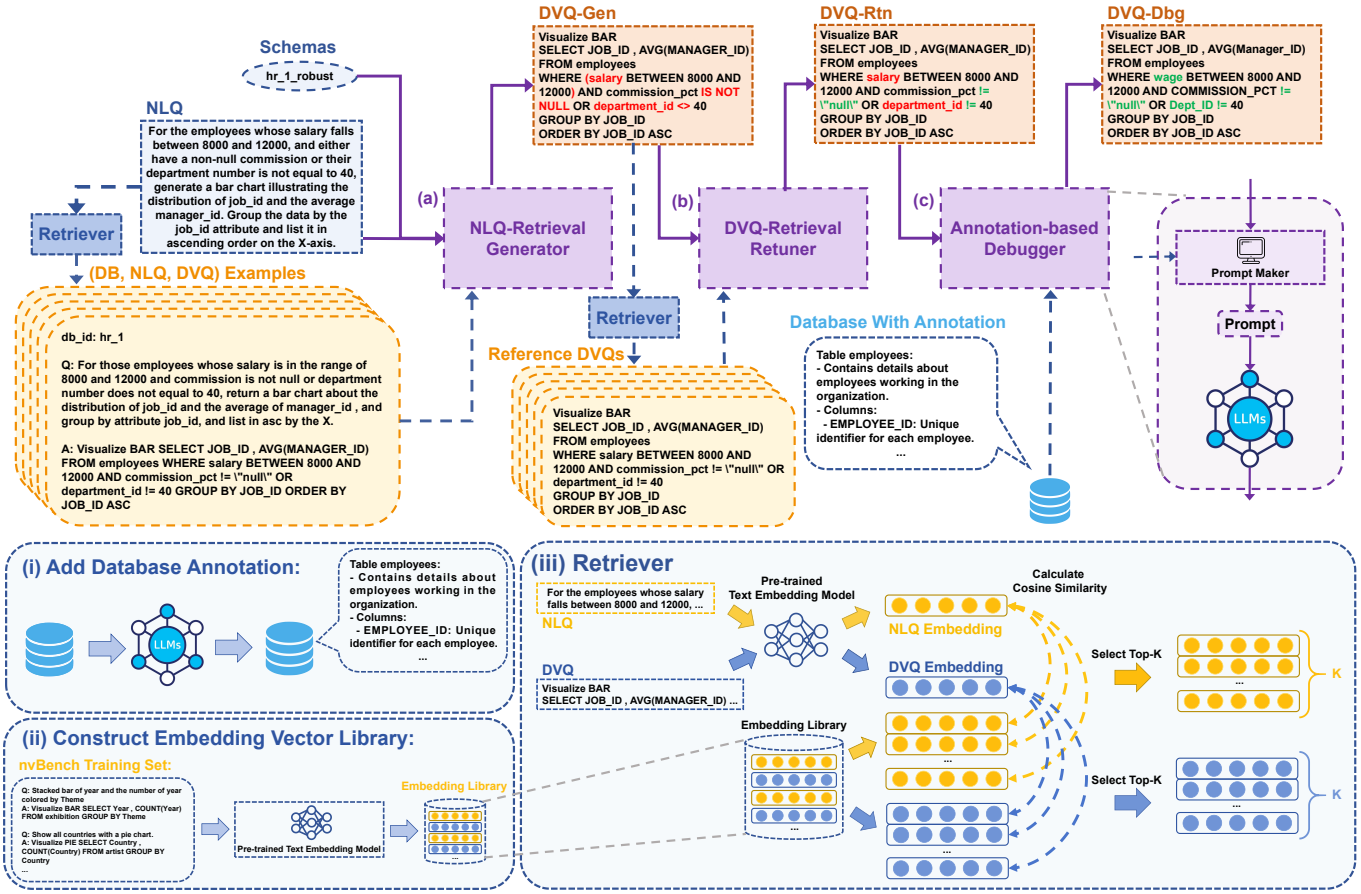


Fig. 5: The working pipeline of our proposed GRED method, which includes three steps: (a) Input the NLQ into the Retriever to obtain the top-K (DB, NLQ, Schemas) instances, then input these instances along with the NLQ and Schemas into the ***NLQ-Retrieval Generator*** to get *DVQ_Gen*; (b) Input the DVQ_Gen into the Retriever to obtain the top-K DVQs, referred to as Reference DVQs, then input Reference DVQs along with DVQ_Gen 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*.

Before all the main processes of GRED, there are some preparatory works that need to be completed.

A. 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.

B. 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 V-A, and then calculates their cosine similarity with the embedding vectors of NLQs in the embedding vector library constructed during the preparation. After that, it selects the top-K most similar NLQs and assembles their corresponding examples into a generation prompt. This prompt is then input into LLM like GPT-3.5-Turbo to generate the corresponding DVQ, referred to as DVQ_{gen}. This approach allows the LLM to achieve more accurate results based on the examples, thus reducing the model's hallucinations.

```
#### Given Natural Language Questions,
Generate DVQs based on their correspoding
Database Schemas.
{Top-K_Examples}
### Database Schemas:
{Schemas_str}
### Chart Type: [ BAR , PIE , LINE , SCATTER ]
### Natural Language Question:
# ''{NLQ}''
### Data Visualization Query:
Visualize
```

The above is the prompt template used in the NLQ-Retrieval Generator. By using structured text expression and clear context organization, the task objectives for the LLM and the exemplary cases to be referenced when completing the task are clearly defined.

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 V-A. 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"*".

The following is the prompt template used in the DVQ-Retrieval Retuner. This template is similar to the structured expression in the NLQ-Retrieval Generator. However, the difference lies in the fact that the purpose of the DVQ-Retrieval Retuner is solely to address the issue of diverse DVQ styles in the dataset. Therefore, in this step, it is only necessary to provide the reference DVQ and the DVQ_{gen} that needs to be modified.

```
### Reference DVQs:
{Top-K_DVQs}
#### Given the Reference DVQs, please modify
the Original DVQ to mimic the style of the
Reference DVQs.
### Original DVQ:
# {DVQ_Rtn}
A: Let's think step by step!
```

Annotation-based Debugger. The examples in the embedding vector library constructed in Section V-A 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}.

```
#### Please generate detailed natural language
 annotations to the following database schemas
### Database Schemas:
{Schemas_str}
### Natural Language Annotations:
{Annotation_of_DB}
#### Given Database Schemas and their
corresponding Natural Language Annotations,
Please replace the column names in the Data
Visualization Query that do not exist in the
database.
### Original DVQ:
# {Original_DVQ}
A: Let's think step by step!
```

Algorithm 1: GRED Algorithm

Inputs : NLQ list in Test Set $\mathcal{Q}$;
Schema list in Test Set $\mathcal{S}$;
NLQ list in Training Set $\mathcal{Q}'$;
DVQ list in Training Set $\mathcal{D}'$
Output: DVQ list $\mathcal{D}$

Procedure GRED($\mathcal{Q}$, $\mathcal{S}$):
  // Preparatory Phase
  $\mathcal{A}, \mathcal{V} \leftarrow \{\}$
  for each $db \in \mathcal{S}$ **do**
    // generate annotation of db
    $\mathcal{A}$.append(GenAnn(db))
  for each $(q', d') \in (\mathcal{Q}', \mathcal{D}')$ **do**
    // text-to-embedding
    $\mathcal{V}$.append(GenEmb(q'), GenEmb(d'))
  // Pipeline of GRED
  $\mathcal{D} \leftarrow$ []
  for each $(q, s) \in (\mathcal{Q}, \mathcal{S})$ **do**
    // NLQ-Retrieval Generator
    $\mathcal{Q}_{\text{ref}} \leftarrow$ Retrieve-By-NLQ(q, $\mathcal{V}$)
    $\mathcal{P}_{\text{gen}} \leftarrow$ Prompt-Maker(q, s, $\mathcal{Q}_{ref}$)
    $\mathcal{D}_{\text{gen}} \leftarrow$ Call-LLM($\mathcal{P}_{gen}$)
    // DVQ-Retrieval Retuner
    $\mathcal{D}_{\text{ref}} \leftarrow$ Retrieve-By-DVQ($\mathcal{D}_{gen}$, $\mathcal{V}$)
    $\mathcal{P}_{\text{rtn}} \leftarrow$ Prompt-Maker($\mathcal{D}_{gen}$, q, s, $\mathcal{D}_{ref}$)
    $\mathcal{D}_{\text{rtn}}$.append(Call-LLM($\mathcal{P}_{rtn}$))
    // Annotation-based Debugger
    $\mathcal{P}_{\text{rtn}} \leftarrow$ Prompt-Maker($\mathcal{D}_{rtn}$, s, $\mathcal{A}$[Match(s)])
    $\mathcal{D}$.append(Call-LLM($\mathcal{P}_{rtn}$))
  return $\mathcal{D}$

The prompt template used in the Annotation-based Debugger is as above. This step's template does not use NLQ because the words in the NLQ might cause hallucinations in the LLM, leading it to mistakenly believe that the incorrect schemas in the DVQ are correct (since the schema matches the words in the NLQ). In reality, these schemas are stored in the database in the form of their synonyms. For example, 'date_of_hire' mentioned in the NLQ actually refers to the 'hire_date' column in the database.

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.

The algorithm for the complete GRED process is represented as Algorithm 1. GRED requires all cases from the training and test sets, namely $\mathcal{Q}'$ and $\mathcal{D}'$ (the NLQ list and DVQ list from the training set) and $\mathcal{Q}$ and $\mathcal{S}$ (the NLQ list and Schemas list from the test set). During the preparation phase, it constructs the annotation set $\mathcal{A}$ for the database and uses $\mathcal{Q}'$ and $\mathcal{D}'$ from the training set to build a retrieval library V for supplementing external knowledge to the LLM. Subsequently, in the formal workflow of GRED, for each example in $\mathcal{Q}$ and $\mathcal{S}$, it performs the steps of *NLQ-Retrieval Generator*, *DVQ-Retrieval Retuner*, and *Annotation-based Debugger*. The results obtained are added to the final result list $\mathcal{D}$. Finally, $\mathcal{D}$

is returned, and the workflow ends.

VI. 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.

A. 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 evaluated GRED and previous text-to-vis models on nvBench-Rob, with the baseline models as follows:

- **Seq2Vis**: Adopting the Seq2Vis model from ncNet [5] as a baseline, Seq2Vis is a sequence-to-sequence model comprising an encoder and a decoder.
- **Transformer**: Utilizing the Transformer model from [44] which is primarily constructed on attention mechanisms, enabling it to understand the input effectively and generate the output.
- **RGVisNet**: RGVisNet [6] utilizes a hybrid retrieval-revision network structure to retrieve the code repository for relevant code based on the NLQ and table schemas, and then revises it accordingly, leading to high-performance outcomes.
- **Few-Shot LLM**: Few-Shot Prompting is an important technique in in-context learning. By providing a few examples in the context, it enables LLMs to understand and follow the instruction.
- **Prompt4Vis**: Prompt4Vis [45] is currently the state-of-the-art (SOTA) solution for the text-to-vis task. This method is mainly divided into three processes: Example Mining, Schema Filtering, and the final generation process by the LLM.
- **Self-correct LLM**: To introduce a more powerful LLM framework based on the self-correct mechanism as a baseline model, we designed an LLM framework that performs self-correction mechanism based on execution results of DVQ, inspired by self-debug [46]. This framework employs the same DVQ generation module as GRED and uses a specially designed execution-based debugger to correct the DVQ.
- **Fine-tuned Llama**: To address the issue of DNN-based models experiencing a sharp decline in performance when encountering unknown tokens, we specifically fine-tuned a small LLM to represent the performance of the model training approach when dealing with NLQ variants and database schema variants. The small LLM we used is `Llama-3.2-1B`. The training method employed is full-parameter fine-tuning.
- **GRED**: GRED is a framework proposed in this work, consisting of three components that address the issues of weak robustness in text-to-visualization models caused by NLQ variants, DVQ programming style differences, and Data Schema variants. To fully validate the effectiveness of GRED on other LLMs, we conducted experiments on both `GPT-3.5-turbo` and `Qwen-plus`.

All these baseline models are trained or acquire external knowledge on the training set of the benchmark dataset nvBench, and their performance is tested and detailed robustness analysis is conducted on nvBench-Rob.

Detailed Definitions of the Evaluation Metrics. Following [5], [6], four popular metrics, namely *Vis Accuracy*, *Data Accuracy*, *Axis Accuracy*, and *Overall Accuracy*, are used in our experiment to evaluate the performance. The detailed definitions of these four metrics are as follows.

- ***Overall Accuracy***: This metric measures exact matches between the predicted DV query and the target DV query. The accuracy calculation formula is:
$$\texttt{Acc.} = \texttt{N}_\texttt{c} \;/\; \texttt{N}$$
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:
$$\texttt{Vis Acc.} = \texttt{N}_{\texttt{Vis}} \;/\; \texttt{N}$$
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:
$$\texttt{Axis Acc.} = \texttt{N}_{\texttt{Axis}} \;/\; \texttt{N}$$
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

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%
Few-Shot LLM	97.04%	33.93%	44.92%	22.17%
Prompt4Vis	86.67%	29.63%	35.82%	17.49%
Self-correct LLM	97.88%	45.01%	70.98%	41.29%
Fine-tuned Llama	98.22%	**74.20%**	84.01%	**71.91%**
GRED (qwen-plus)	**98.39%**	58.46%	76.99%	53.47%
GRED (gpt-3.5-turbo)	97.63%	61.93%	**88.41%**	59.98%

TABLE I: 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%
Few-Shot LLM	96.79%	30.29%	31.64%	17.26%
Prompt4Vis	88.06%	28.45%	25.90%	14.84%
Self-correct LLM	97.72%	71.32%	50.08%	45.43%
Fine-tuned Llama	98.05%	57.28%	57.78%	49.41%
GRED (qwen-plus)	**98.48%**	59.90%	81.22%	55.75%
GRED (gpt-3.5-turbo)	97.72%	**65.48%**	**85.03%**	**61.93%**

TABLE II: 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%
Few-Shot LLM	97.38%	30.37%	30.03%	17.18%
Prompt4Vis	85.95%	27.04%	27.66%	13.70%
Self-correct LLM	98.14%	43.97%	68.35%	40.00%
Fine-tuned Llama	98.22%	50.51%	53.05%	41.37%
GRED (qwen-plus)	**98.65%**	53.98%	74.70%	49.24%
GRED (gpt-3.5-turbo)	98.14%	**58.48%**	**81.52%**	**54.85%**

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

calculation formula is:

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

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

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
```

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
```

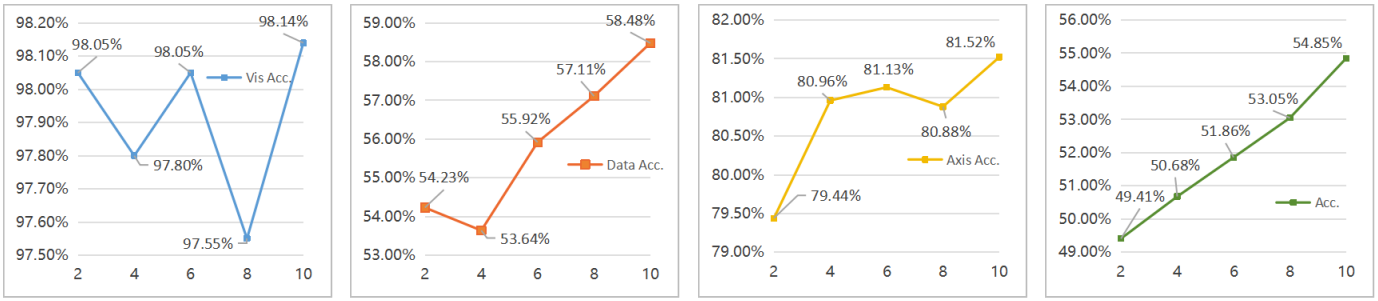
In addition, the large language model used in the experimental process is *GPT-3.5-Turbo* and uses the version released by OpenAI on *January 25, 2024*.

B. Performance Comparison

As shown in Figure 4, 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 Prompt4Vis experiences a significant drop in accuracy.

To comprehensively assess the robustness of the models, we trained all baseline models on nvBench and tested them on the three test sets of nvBench-Rob. The results are shown in Table I, Table II and Table III. The previous SOTA model in the text-to-vis field, Prompt4Vis, achieved an accuracy of 70.73% on the randomly split nvBench dataset, but its performance dropped sharply when facing variations, with accuracy decreasing by 53.24% or 55.89% on single-variant datasets and by 57.03% on the dual-variant dataset. Due to our random split method (rather than cross-domain split), the training set included most of the databases, leading to the DNN-based model RGVisNet outperforming the previous SOTA model after training (85.17% vs. 70.73%). Nevertheless, RGVisNet's accuracy still significantly dropped on the single-variant test sets, by 39.3% and 40.26%, respectively. The most notable difference was observed on nvBench-Rob$_{(nlq,schema)}$, where the accuracy dropped by 60% compared to the original nvBench test set. Meanwhile, GRED demonstrated impressively high accuracy across the three test sets of nvBench-Rob, with improvements of 42.49% and 47.09% on the single-variant test sets compared to the previous SOTA model Prompt4Vis, and an improvement of 41.15% on the most challenging dual-variant test set. These results indicate that GRED has a strong ability to resist interference with model inputs, showcasing its excellent robustness.

Additionally, after a detailed analysis of the baseline models, we found that model training-based methods and Prompting LLM-based methods exhibit different performances when facing NLQ variants and data schema variants. As shown in Table I and II, we used Fine-tuned Llama as a representative of model training methods because it is not surprising that model training methods like RGVisNet experience significant performance drops when encountering sequences and tokens not present in the training set (this is their out-of-domain knowledge). We used GRED as a representative of Prompting LLM-based methods. We found that model training-based methods showed impressive performance when facing NLQ variants alone, achieving the best accuracy (although it dropped by 18.71% compared to its accuracy on nvBench). However, their performance significantly declined when facing data schema variants, with accuracy dropping by 42.21%. This could be due to the strong natural language understanding (NLU) capabilities of LLMs, which can establish good mappings between NLQ variants and data schemas in the training corpus, but fail to correctly execute the schema linking process when



(a) Vis Acc. (b) Data Acc. (c) Axis Acc. (d) Overall Acc.

Fig. 6: Parameter Study on nvBench-Rob$_{(nlq,schema)}$. The vertical axis represents accuracy, and the horizontal axis represents the number of retrieved examples.

encountering data schemas not present in the training corpus. On the other hand, the performance of Prompting LLM-based methods was more stable, with accuracy dropping by 8.72% when facing NLQ variants alone and by 6.77% when facing data schema variants alone. This indicates that Prompting-based methods are more robust than model training-based methods and can still establish correct mappings when facing entirely new schema linking scenarios.

Additionally, to fully validate the advanced nature of the GRED framework, we introduced Few-Shot LLM and Self-correct LLM as baseline models. The low accuracy performance of Few-Shot LLM (with accuracies of 22.17%, 17.26%, and 17.18% on the three test sets of nvBench-Rob, respectively) demonstrates that GRED's strong performance does not stem from the inherent capabilities of the underlying LLM, but rather from the specially designed multi-step execution framework. Furthermore, the Self-correct LLM, which generates code based on RAG technology and then self-corrects according to the DVQ execution results, showed a decrease in accuracy of over 15% on all test sets of nvBench-Rob compared to GRED. This proves the advanced robustness of GRED in the text-to-vis field.

Finally, based on our analysis of GRED (gpt-3.5-turbo) and GRED (qwen-plus) on nvBench-Rob, we found that even when using qwen-plus, GRED still achieves the second-highest accuracy on the most challenging test set nvBench-Rob$_{(nlq,schema)}$, just behind GRED (gpt-3.5-turbo). This validates that GRED is applicable to non-GPT models, demonstrating the generalizability of GRED.

C. Parameter Study

To study the impact of parameter changes on the performance of GRED, we conducted another's performance is the number of retrieved examples in the RAG technique. To best reflect the model's performance when generalizing to real-world scenarios, we used nvBench-Rob$_{(nlq,schema)}$) as the test set and employed the evaluation metrics described in Section VI-A. The experimental results are shown in Figure 6. We obtained four subfigure, each describing the accuracy curves of GRED under four different evaluation metrics as the number of retrieved examples varies. It can be observed that as the number of reference examples increases, the performance of GRED in Vis Acc., Axis Acc., and Data Acc. shows a certain decline when the number of retrieved examples is `4` or `8`. However, for the Overall Acc., which best demonstrates the model's performance, there is a stable upward trend. Among the four evaluation metrics, the accuracy is highest when the number of retrieved examples is **`10`**. Since further increasing the number of retrieved examples would result in a prompt context length that is too long to be input into the LLM, we could not conduct experiments with more retrieved examples.

D. 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 the Annotation-based Debugger **(w/o DBG)**; (iii) removing the DVQ-Retrieval Retuner **(w/o RTN)**.

Additionally, we also evaluated the Few-shot Prompting method and Self-correct LLM for the LLM. To ensure a fair comparison, the number of examples in the few-shot scenarios was kept consistent with those in GRED, to verify that each component in GRED outperforms the direct Few-shot Prompting method. In the Self-correct LLM method, we also used a RAG-based DVQ generator. The difference is that the debugger we used corrects the code based on the execution results of the DVQ, to verify the advanced nature of the retuner and debugger in GRED.

The ablation study results shown in Table IV confirm the importance of the three design components in our proposed model. We observed that the NLQ-Retrieval Generator plays a crucial role in countering input perturbations caused by NLQ 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 because it helps the LLM adjust the generated DVQ style to better match the dataset's style, thereby reducing

Model	nvBench-Rob$_{nlq}$	nvBench-Rob$_{schema}$	nvBench-Rob$_{(nlq,schema)}$
Few-shot LLM	22.17%	17.26%	17.18%
Prompt4Vis (SOTA)	17.49%	14.84%	13.70%
Self-correct LLM	41.29%	45.43%	40.00%
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 DBG	61.68%	42.47%	38.57%

TABLE IV: Ablation Study Results on nvBench-Rob. This table shows the performance of GRED on nvBench-Rob after removing each component. Additionally, to demonstrate the advancement of each component in GRED, we added Few-Shot LLM and Self-correct LLM as supplementary experiments.

errors in programming style and achieving higher accuracy. Therefore, these three components all contribute to the model's robustness. In contrast, the Few-shot LLM performs worse than all experimental settings of GRED across all three test sets. This discrepancy underscores that each component in GRED outperforms the direct Few-shot Prompting method. The Self-correct method performs better than the w/o DBG setting when facing data schema variations but significantly worse than GRED and the w/o RTN setting, indicating that the Annotation-based Debugger component is better at handling data schema variability compared to the Execution-based Debugger. Additionally, the Self-correct LLM performs significantly worse than all experimental settings of GRED when facing NLQ variants. Thus, the advanced nature of the retuner and debugger in GRED is validated.

E. Case Study

Table V presents a case study illustrating the DVQ generated by GRED and all baseline models, along with the final DV charts. As shown in the table, Seq2Vis generates incorrect column names and aggregation keywords on the y-axis, resulting in no legend being displayed in the chart in Table Vb. RGVisNet and Transformer generate DVQs with the correct aggregation keywords. However, due to a 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 Vc. The previous SOTA model Prompt4Vis is severely affected by NLQ and data schema variations, resulting in a DVQ that selects a chart type not conforming to DVQ syntax and misinterprets the data on the XY axes. Due to these syntax errors, Prompt4Vis ultimately fails to produce a chart. Fine-tuned Llama is more susceptible to NLQ and data schema variations, selecting a non-existent data schema ("Dpt_ID") and severely affecting data operations, ultimately failing to generate a chart. Self-correct LLM fails to correctly understand the XY axis data and data operations, resulting in a chart with no legend(as shown in Table Vd). Unlike the aforementioned models, GRED is capable of not only generating a structure identical to the target query but also producing the correct column names, thereby resulting in accurate charts as shown in Table Va.

F. Summary

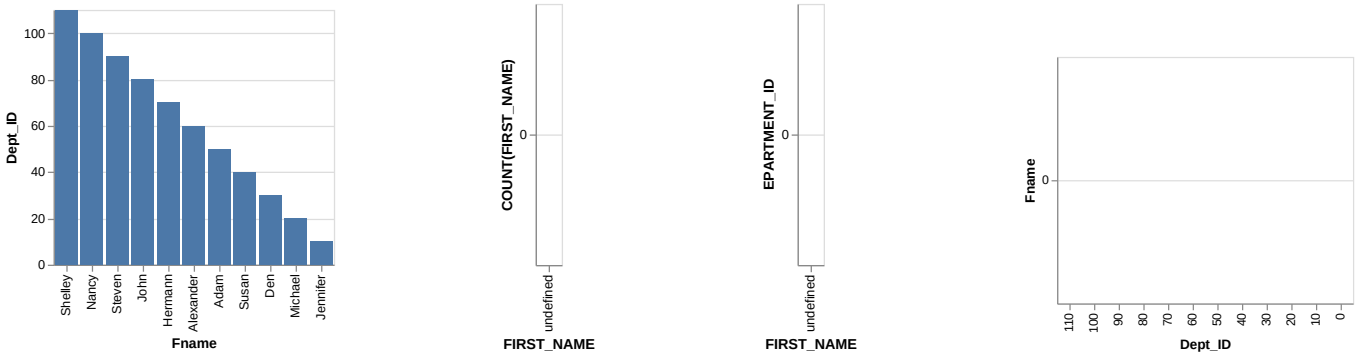
The experimental results demonstrate that: (i) Previous text-to-vis models, after being trained on the benchmark dataset nvBench, fail to generalize to real-world contexts, as evidenced by a significant drop in accuracy on the robustness test set nvBench-Rob in comparison to their performance on the benchmark test set. (ii) The study introduces a framework named GRED, specifically designed to address the poor generalization capabilities of text-to-vis models by leveraging RAG technology. Through extensive experimentation, GRED has demonstrated superior performance compared to previous text-to-vis models in terms of generalization, effectively addressing challenges encountered in real-world scenarios. Specifically, GRED exhibits proficiency in handling inquiries from non-technical users who may not explicitly mention table schemas or DVQ syntax keywords and may present questions in various formats. Additionally, GRED demonstrates efficient schema linking capabilities when faced with databases created under diverse naming conventions. (iii) All components within the GRED framework are effective; optimal accuracy is achieved only when all components collaboratively participate in generating the DVQ code. (iv) The superior performance of the GRED framework is not solely attributed to the inherent powerful natural language understanding (NLU) capabilities of LLM itself; instead, it primarily arises from the advanced framework design. (v) GRED is versatile and applicable to both GPT and non-GPT models. In summary, we introduced nvBench-Rob to highlight the shortcomings of previous text-to-vis models in terms of robustness. In response, we developed GRED, which delivers optimal performance in real-world settings, boasting a 30% improvement in accuracy over past SOTA models.

Due to space limitations in this paper, the comparative analysis of GRED with non-RAG methods (such as data augmentation), the detailed evaluation of each method's performance under specific metrics (Vis Acc., Data Acc., and Axis Acc.) when facing NLQ and data schema variations , error analysis of examples that GRED Fails and more robustness cases analysis can be found in the online technical report provided in the footnote of Section I.

VII. Conclusion and Future Work

Robustness is a crucial factor for evaluating model performance. In this study, we introduce the first comprehen-

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 Va*
Seq2Vis	Visualize BAR SELECT FIRST_NAME , COUNT (FIRST_NAME) FROM dogs ORDER BY COUNT (LAST_NAME) DESC → *Figure Vb*
Transformer	Visualize BAR SELECT FIRST_NAME , DEPARTMENT_ID FROM employees ORDER BY DEPARTMENT_ID DESC → *Figure Vc*
RGVisNet	Visualize BAR SELECT FIRST_NAME , DEPARTMENT_ID FROM employees ORDER BY DEPARTMENT_ID DESC → *Figure Vc*
Prompt4Vis	Visualize HISTOGRAM SELECT Dept_ID , Fname FROM employees ORDER BY Dept_ID DESC → *"KeyError: 'HISTOGRAM"'*
Self-correct LLM	Visualize BAR SELECT Dept_ID, Fname FROM employees GROUP BY Dept_ID, Fname ORDER BY Dept_ID DESC → *Figure Vd*
Fine-tuned Llama	Visualize BAR SELECT T1.Fname, T1.Dpt_ID FROM employees AS T1 JOIN departments AS T2 ON T1.department_id = T2.department_id WHERE T1.employee_id = T2.manager_id ORDER BY T1.Dpt_ID DESC → *"sqlite3.OperationalError: no such column: T1.Dpt_ID"*
GRED	Visualize BAR SELECT **Fname , Dept_ID** FROM employees ORDER BY **Dept_ID** DESC → *Figure Va*



(a) ✔ (b) ✘ (c) ✘ (d) ✘

TABLE V: Case Study. DVQs generated by other baselines like Prompt4Vis and GRED, together with their corresponding visualization charts (Errors are marked with red colors).

sive 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.

Although nvBench-Rob provides a comprehensive robustness evaluation of text-to-vis models through NLQ variations and in nvBench-Rob is insufficient to fully simulate the variety of NLQs posed by different groups in real-world scenarios. To further optimize nvBench-Rob, we will refer to existing diversity research [47] for the next steps of improvement.

Additionally, considering the efficiency issues in real-world scenarios, we will further explore the combination of small language models (SLM, such as Llama-3.2-1B) with the GRED method. For instance, we will use retrieval-augmented fine-tuning (RAFT) [48] techniques to enable the small language model to fully utilize the effective information from retrieved examples, thereby achieving high performance while maintaining high efficiency.

ACKNOWLEDGMENT

Yuanfeng Song and Haodi Zhang are the corresponding authors. Yuanfeng Song and Raymond Chi-Wing Wong are supported by fund WEB24EG01-H. Haodi Zhang is supported by the fund 2022A1515010675. Chen Zhang is supported by the following funds: P0036742, P0038989, P0040041, P0040568, P0043864, P0045948, P0046453, P0046701, P0046703, P0048183, P0048566, P0048191 and P0048887.

REFERENCES

[1] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega-lite: A grammar of interactive graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, pp. 341–350, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:206805969

[2] V. Gómez-Rubio, "ggplot2 - elegant graphics for data analysis (2nd edition)," *Journal of Statistical Software*, vol. 77, pp. 1–3, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:1607901

[3] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. G. Parameswaran, "Effortless data exploration with zenvisage: An expressive and interactive visual analytics system," *Proc. VLDB Endow.*, vol. 10, pp. 457–468, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:16887553

[4] D. Li, H. Mei, Y. Shen, S. Su, W. Zhang, J. Wang, M. Zu, and W. Chen, "Echarts: A declarative framework for rapid construction of web-based visualization," *Vis. Informatics*, vol. 2, pp. 136–146, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52076092

[5] Y. Luo, N. Tang, G. Li, C. Chai, W. Li, and X. Qin, "Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks," in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1235–1247. [Online]. Available: https://doi.org/10.1145/3448016.3457261

[6] Y. Song, X. Zhao, R. C.-W. Wong, and D. Jiang, "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*, 2022, pp. 1646–1655.

[7] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[8] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[9] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. Del Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi *et al.*, "Textbooks are all you need," *arXiv preprint arXiv:2306.11644*, 2023.

[10] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[11] OpenAI., "Gpt-4 technical report," 2024.

[12] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," *arXiv preprint arXiv:2004.09813*, 2020.

[13] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic bert sentence embedding," *arXiv preprint arXiv:2007.01852*, 2020.

[14] Y. Ge, J. Wei, Y. Song, C. Zhang, and R. C.-W. Wong, "Automatic data visualization generation from chinese natural language questions," in *The 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, 2024.

[15] X. Qian, R. A. Rossi, F. Du, S. Kim, E. Koh, S. Malik, T. Y. Lee, and J. Chan, "Learning to recommend visualizations from data," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1359–1369.

[16] T. Ho, T. Nguyen, and D. Nguyen, "Visualization support for a user-centered kdd process," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 519–524.

[17] U. M. Fayyad, G. G. Grinstein, and A. Wierse, *Information visualization in data mining and knowledge discovery*. Morgan Kaufmann, 2002.

[18] J. Tang, Y. Luo, M. Ouzzani, G. Li, and H. Chen, "Sevi: Speech-to-visualization through neural machine translation," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 2353–2356.

[19] P. Hanrahan, "Vizql: a language for query, analysis and visualization," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 2006, pp. 721–721.

[20] M. Vartak, S. Huang, T. Siddiqui, S. Madden, and A. Parameswaran, "Towards visualization recommendation systems," *Acm Sigmod Record*, vol. 45, no. 4, pp. 34–39, 2017.

[21] Y. Luo, X. Qin, N. Tang, G. Li, and X. Wang, "Deepeye: Creating good data visualizations by keyword search," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1733–1736.

[22] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer, "Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 438–448, 2018.

[23] A. Narechania, A. Srinivasan, and J. Stasko, "Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 369–379, 2020.

[24] V. Dibia and Ç. Demiralp, "Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks," *IEEE computer graphics and applications*, vol. 39, no. 5, pp. 33–46, 2019.

[25] W. Cui, X. Zhang, Y. Wang, H. Huang, B. Chen, L. Fang, H. Zhang, J.-G. Lou, and D. Zhang, "Text-to-viz: Automatic generation of infographics from proportion-related natural language statements," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 906–916, 2019.

[26] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "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*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3911–3921. [Online]. Available: https://aclanthology.org/D18-1425

[27] H. Chen, J. He, K. Narasimhan, and D. Chen, "Can rationalization improve robustness?" in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 3792–3805. [Online]. Available: https://aclanthology.org/2022.naacl-main.278

[28] J. Yan, Y. Xiao, S. Mukherjee, B. Y. Lin, R. Jia, and X. Ren, "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*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 508–520. [Online]. Available: https://aclanthology.org/2022.naacl-main.37

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

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

[31] Y. Zhao, C. Zhao, L. Nan, Z. Qi, W. Zhang, X. Tang, B. Mi, and D. Radev, "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)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 6064–6081. [Online]. Available: https://aclanthology.org/2023.acl-long.334

[32] X. Chen, G. Long, C. Tao, M. Li, X. Gao, C. Zhang, and X. Zhang, "Improving the robustness of summarization systems with dual augmentation," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 6846–6857. [Online]. Available: https://aclanthology.org/2023.acl-long.378

[33] S. Wang, Z. Li, H. Qian, C. Yang, Z. Wang, M. Shang, V. Kumar, S. Tan, B. Ray, P. Bhatia, R. Nallapati, M. K. Ramanathan, D. Roth, and B. Xiang, "ReCode: Robustness evaluation of code generation models," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 13 818–13 843. [Online]. Available: https://aclanthology.org/2023.acl-long.773

[34] X. Wang, H. Wang, and D. Yang, "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, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 4569–4586. [Online]. Available: https://aclanthology.org/2022.naacl-main.339

[35] G. Yu, L. Liu, H. Jiang, S. Shi, and X. Ao, "Retrieval-augmented few-shot text classification," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 6721–6735. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.447

[36] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 9248–9274. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.620

[37] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "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)*. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 10 014–10 037. [Online]. Available: https://aclanthology.org/2023.acl-long.557

[38] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen, and J. Gao, "Check your facts and try again: Improving large language models with external knowledge and automated feedback," 2023.

[39] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "Chatlaw: Open-source legal large language model with integrated external knowledge bases," 2023.

[40] S. Zhou, U. Alon, F. F. Xu, Z. Jiang, and G. Neubig, "Docprompting: Generating code by retrieving the docs," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=ZTCxT2t2Ru

[41] T. Ren, Y. Fan, Z. He, R. Huang, J. Dai, C. Huang, Y. Jing, K. Zhang, Y. Yang, and X. S. Wang, "Purple: Making a large language model a better sql writer," *arXiv preprint arXiv:2403.20014*, 2024.

[42] L. Dou, Y. Gao, M. Pan, D. Wang, W. Che, D. Zhan, and J.-G. Lou, "Multispider: towards benchmarking multilingual text-to-sql semantic parsing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 12 745–12 753.

[43] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin, "Natural language to visualization by neural machine translation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 217–226, 2021.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[45] S. Li, X. Chen, Y. Song, Y. Song, and C. Zhang, "Prompt4vis: Prompting large language models with example mining and schema filtering for tabular data visualization," 2024. [Online]. Available: https://arxiv.org/abs/2402.07909

[46] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," 2023. [Online]. Available: https://arxiv.org/abs/2304.05128

[47] T. Ge, X. Chan, X. Wang, D. Yu, H. Mi, and D. Yu, "Scaling synthetic data creation with 1,000,000,000 personas," 2024. [Online]. Available: https://arxiv.org/abs/2406.20094

[48] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, "Raft: Adapting language model to domain specific rag," *arXiv preprint arXiv:2403.10131*, 2024.