FeVisQA: Free-form Question Answering over Data Visualizations

Yuanfeng Song*, Jinwei Lu†, Yuanwei Song¶, Caleb Chen Cao‡, Raymond Chi-Wing Wong‡, Haodi Zhang†§
*WeBank AI, Shenzhen, China †Shenzhen University, Shenzhen, China
‡HKUST, Hong Kong, China §HKUST-GZ, Guangzhou, China ¶Huawei Research, Hong Kong, China

***Abstract*—Given a massive dataset, *data visualization* (DV) could efficiently express the insights and summaries behind the massive raw data by employing vivid visual representations. To create suitable DVs, users are required to get a comprehensive understanding of the raw data and then transfer their ideas into DVs by composing a suitable and accurate *specification* in some *declarative visualization languages* (DVLs, e.g., Vega-Lite). A specification is a JSON object defining the properties of the DVs, like the selected data, the transformations, the visual details, and so on. Due to its complicated grammar and details, DV has quite a steep learning curve, even for data analysts. In this paper, we propose a new task named *FeVisQA*, referring to Free-form Question Answering over data Visualizations. More specifically, given a raw dataset, a related DV (in the form of a specification), and a question, FeVisQA aims to predict a textual answer automatically. As a particular case of the general CodeQA (i.e., QA over general programming code like Python and Java) task, FeVisQA enables people to better comprehend data and its DVs by conducting logical reasoning when answering these questions. Since FeVisQA has not been studied in the literature, we first construct a benchmark dataset containing 152 datasets, 14,406 DVs, and 83,890 QA pairs. To tackle this new task, we design a novel neural network named *FeVisQANet* with advanced multi-modal encoder and adaptive decoder structures, and we also design a novel multi-step framework called *VisQA* for Multi-modal Large Language Models (MLLMs) based on Retrieval-augmented Generation (RAG) technology. Extensive experiments on our constructed datasets validate the rationale and effectiveness of this proposed FeVisQA task and the proposed model. While research on QA over text and table, machine reading comprehension, and CodeQA develops rapidly, prior works have yet to draw attention to question-answering over DVs. This study connects two important subareas, QA from the natural language process area and DV from the data engineering area. We hope this new dataset and model can serve as a helpful benchmark that would benefit the development of both fields.**

***Index Terms*—Question-Answering, CodeQA, Data Visualization, Vega-Lite, FeVisQA, Declarative Visualization Languages**

I. Introduction

Data Visualizations (DVs), following the adage *"A picture is worth a thousand words"*, effectively convey insights from massive datasets [2], [3]. Recently, a significant amount of studies related to DVs has been extensively explored and published in data engineering and database conferences such as SIGMOD [4], VLDB [5], [6], TKDE [7]–[9] and ICDE [10]–[14]. However, when facing a massive raw dataset, to create suitable DVs, users need to define a visualization *specification* in some *Declarative Visualization Languages* (DVLs, e.g., Vega-Lite [15]). A specification is usually in the form of a JSON object that follows the grammar of some DVLs. It includes many details and properties like the selected data, the transformations, and the visual details. Fig. 1 shows an example of a car dataset with a specification in the widely-used Vega-Lite DVL. It should be noted that the specification could be complicated with many details. To create good DVs, users must not only understand their raw data deeply, but also become experts in DVLs. The users must be very familiar with the DVLs and, at the same time, achieve a deep understanding of the raw data, to create feasible and suitable specifications. This is just like how a programmer has to master a new programming language to write code. These premises are like how a programmer masters a new programming language.This steep learning curve prevents many people from using and understanding data visualizations effectively, limiting the power of DVs to help everyone learn from data.

To make data visualizations more accessible and understandable for everyone, not just experts, in this work, we

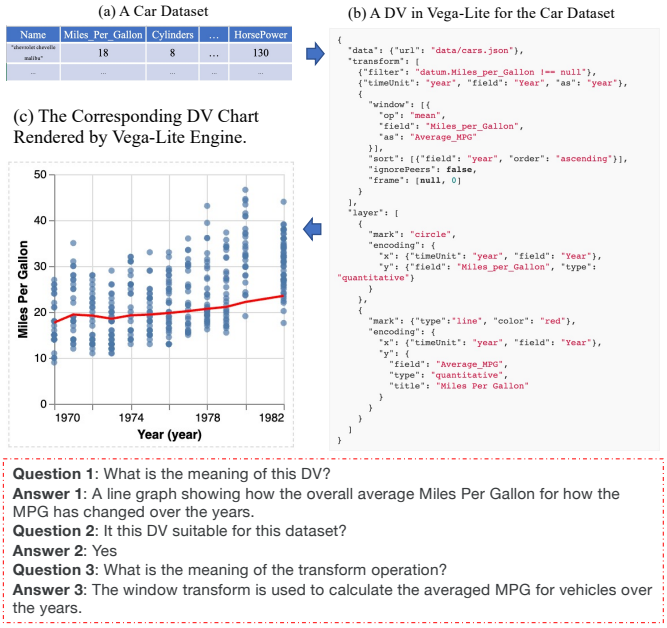


Fig. 1: Examples showing three free-form questions over a DV (Specification) in Vega-Lite and their corresponding answers in the proposed FeVisQA task.

A FeVisQASystem system using the techniques described in this draft was published as a demonstration paper at ICDE 2024 [1].

propose a new task named **FeVisQA**, referring to Free-form Question Answering over data Visualization. More specifically, given a DV specification of a dataset and a question, the objective of the FeVisQA task is to predict the corresponding textual answer. FeVisQA could be considered to be a particular case of the general CodeQA tasks [16], [17], which focus on answering questions related to programming codes for source code comprehension. Serving a similar goal of facilitating programming learning in education, FeVisQA enables people to better comprehend DVLs by conducting logical reasoning when answering questions. To better illustrate, three QA examples over a DV specification in FeVisQA are also shown in Fig. 1. By answering these reasoning questions, people could better understand the data and the specification since these questions usually require significant perceptual and cognitive effort to obtain the final answers.

Since FeVisQA has not been studied in the literature, we first construct a benchmark dataset to evaluate its rationale and further promote the following research and development of this field. This dataset is composed of various questions that belong to three categories. The first kind concerns a given DV specification's general information (e.g., meaning). The second kind justifies whether a DV specification is good or bad for a given dataset. The third kind is about the details of some given specifications.

After constructing the dataset, we must design models to handle this task. We notice that CodeQA is a well-formed task with a relatively long history. It has received unprecedented attention recently, with many CodeQA models released from the research community. For example, Liu *et al.* [16] proposed a CodeQA dataset that contains QA pairs on general programming languages (GPLs) like Java and Python. Then they further designed several baseline methods to tackle this CodeQA task. Lee *et al.* [17] released another CS1QA dataset for educational purposes that include QA pairs with the codes from students and the portion of the code relevant to answering the question. While most of this effort has been spent on the QA over GPLs, studies have yet to be conducted on QAs dedicated to DVs, and DVLs have a distinctive nature that differs from GPLs. To this end, FeVisQA aims to promote understanding DVs (in terms of *specification* in DVLs). The input of our model is the data, the DV (specification), and the question, while none of the existing QA models achieve this objective. Furthermore, rich information is encoded in the multi-modal inputs in DV, and how to fully utilize them is still an open question.

To alleviate the above-mentioned problems, we present a multi-modal neural network named **FeVisQANet** for answering questions over DVs. FeVisQANet is equipped with a novel multi-model encoder and an advanced adaptive decoder. In particular, we first incorporate the dataset and the question information by a dataset encoder. To better preserve the structural and multi-modal information from DV specifications, we represent each DV specification into an Abstract Structural Tree (AST), which follows a tree-like grammar that preserves structural and contextual information. Then we employ a Transformer-based structure to encode the AST information into node embeddings. To generate accurate answers to different kinds of questions, we further propose an adaptive decoder to predict answers like long-form strings, numbers, and boolean values. Extensive experiments over our constructed dataset validate that our method could considerably improve other strong baselines.

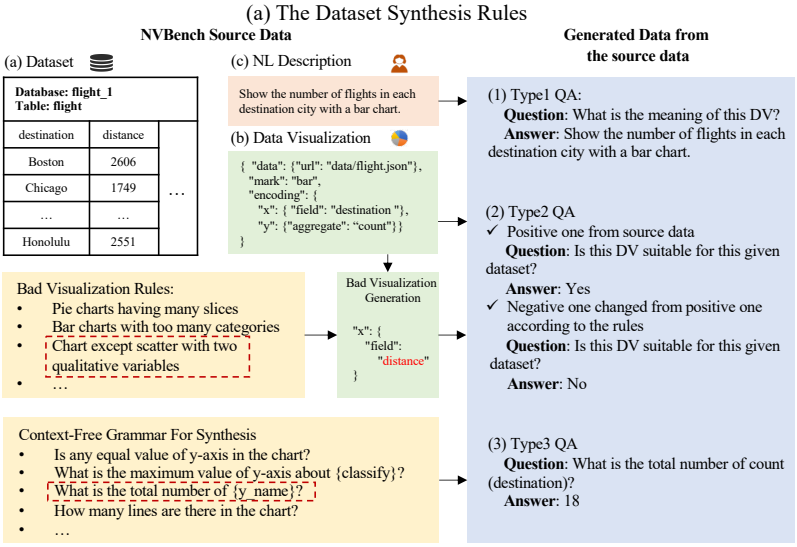
To further explore the application of multimodal large language models (MLLMs) in the FeVisQA task, we designed the VisQA framework, a multi-step framework that includes Question Classification, Initial Answer Generation, RAG(Retrieval-augmented Generation)-based Reasoning, and Answer Validation and Correction. Specifically, for each natural language question input, we (i) use a FeVisQANet-based classifier to categorize whether it is necessary to retrieve relevant examples as contextual supplements and whether to execute DVL to produce a DV chart as multimodal input; (ii) construct a prompt based on the classification results, which includes the possible example retrieval process and the DVL execution process, and input the prompt into the LLM to obtain the initial answer; (iii) retrieve relevant examples (including question, DVL, and answer) based on the cosine similarity of the input question and DVL, allowing the LLM to generate the reasoning process for answering the question by referencing the answers; (iv) use the reasoning process of the relevant examples as guidance for the LLM to validate and, if necessary, correct the initial answer.

Comparative evaluation of FeVisQANet and VisQA against other baseline models demonstrates their superior performance, achieving state-of-the-art (SOTA) results. Among them, FeVisQANet performs slightly lower than VisQA but still significantly outperforms other baselines, making it highly suitable for resource-constrained scenarios.

This study is a good exploration that marries two important subareas: QA from the natural language processing (NLP) area and DV from the data mining area. We hope it will inspire more cross-area research studies. In a nutshell, our main contributions in this paper are summarized as,

- Being the first to propose this novel FeVisQA [1] task, which focuses on Free-form Question Answering over Data Visualizations. To validate the rationale of this new task, we construct a benchmark dataset consisting of different questions. This new task could help people better comprehend the data and master how to create suitable DVs, given massive raw data. This study can serve as a useful research benchmark that significantly enlarges the QA family.
- We design a novel neural network named FeVisQANet for this new proposed task. FeVisQANet has advanced encoders that can fully use the multi-modal information expressed in DVs. Moreover, it employs an adaptive decoder to generate various answers to questions of

[1]Our source code and FeVisQA dataset are available at https://1drv.ms/f/c/80e862c5ba9a0a46/EjPn1aYVMb5Mg5dbpd4jRRkBw2jHtdhMeWnpoa3UmWUlEw?e=8gzWDd



(b) The Statistics of the Proposed Datasets

	Train	Val.	Test	Total
No. of Databases	106	15	31	152
No. of QAs	61396	7614	14880	83890
No. of DVs	10581	1310	2515	14406
No. of Type1 QA	5294	655	1298	7247
No. of Type2 QA	10574	1310	2434	14318
No. of Type3 QA	45522	5649	11073	62244

(c) The Distrubtion of the Quesions

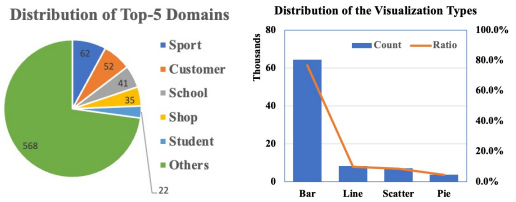


Fig. 2: The methods and data sources used to construct the FeVisQA dataset, and the statistic of our constructed dataset.

different kinds. These new structures guarantee the final inference performance of the whole network.

- We designed a multi-step LLM-based framework, VisQA, which leverages RAG technology and a DNN-based classification model to fully utilize the capabilities of MLLMs in the FeVisQA task.
- We conducted extensive experiments on our proposed dataset and compare it with other strong baselines. The results show that our proposed FeVisQANet model and VisQA framework could perform better than several strong counterparts.

The rest of this paper is organized as follows: we first introduce some concepts and the problem definition in Section II. Then we discuss the details of our proposed FeVisQANet mechanism, including the data creation process and the network structure in Section III. The experimental setup and results are listed in Section V, followed by the related work in Section VI. Finally, we conclude the work in Section VII.

II. Problem Formulation and Datasets

In this section, we first introduce some preliminary concepts that could improve the understanding of the following work. These concepts can be divided into three categories, DV, the proposed FeVisQA task, and the released datasets.

A. Concepts on Data Visualization

Declarative Visualization Language (DVL). A DVL defines the properties of a DV, including features like chart type, color, size, mapping function, and properties for marks such as canvas size and legend. Commonly used DVLs in the market include Vega-Lite [15], ggplot2 [18], ECharts [19], and VizQL [20]. Our analysis mainly uses the Vega-Lite due to its popularity and wide usage. However, the proposed methodology could also be used in other DVLs.

Visualization Specification. A specification defines the exact properties of a DV using a DVL. In Fig. 1, the specification in Vega-Lite follows the JSON format defining properties such as data path, mark, and encoding. A specification could be rather complicated, and creating a suitable specification for a given dataset requires experience in DVL as well as familiarity with the raw data.

B. The Proposed FeVisQA Task

FeVisQA is short for Free-form Question Answering over data Visualization. More specifically, given a dataset/database D, a DV v (in the form of a specification in a DVL) and a question q, the FeVisQA task aims to predict the textual answer a automatically. FeVisQA enables users to better comprehend DVs by conducting logical reasoning when answering these questions. Then, the complete training set can be represented as $\mathcal{T} = \{D^{(o)}, v^{(o)}, q^{(o)}, a^{(o)}\}_{o=1}^{N}$, where N is the dataset size. The desired FeVisQA model could be represented as $f(D, v, q) \rightarrow a$.

C. The Constructed FeVisQA Dataset

Since FeVisQA has not been studied in the literature, we construct some benchmark datasets to promote the development of this new field. As shown in Fig. 2(a), we employ various rules and data sources to construct question-answer pairs and then accumulate a large-scale FeVisQA dataset. The questions can be categorized into different styles, namely *Data Retrieval*, *Reasoning*, and *Structure*. For the reasoning question, we mainly obtain the questions from two sources. (i) the NVBench dataset from a paralleled task, text-to-vis, contains around 10K natural language description (NLD) and their corresponding DVs. For each DV in NVBench, we could define a theme question like *"What is the meaning of this DV?"*, and obtain its answer by modifying the NLD. An example can be found in Fig. 1. This kind is refereed as *Type 1* in the following discussion. (ii) the training dataset from another paralleled task, DV recommendation, which contains around 10K datasets and their positive and negative recommended DVs, given the potential datasets. For each DV

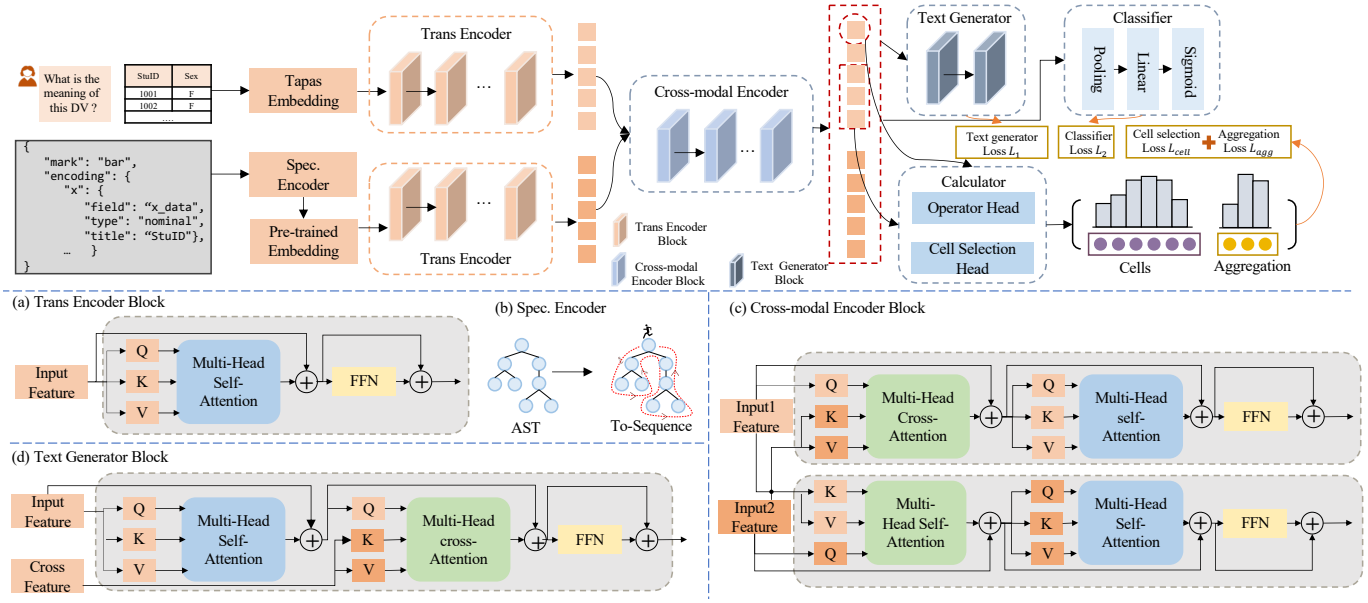


Fig. 3: Network Structure of our Proposed FeVisQANet Model, where (i) a *Question & Data Encoder* first loads Tapas embeddings for words and then encodes the dataset and question information using a Transformer-based structure, (ii) a *Specification Encoder* first converts the DV specification into AST and then encodes it with a GCN structure, (iii) a *Cross-modal Encoder* fuses the outputs from the previous two encoders, and (iv) an *Adaptive Decoder* generates corresponding answers to different question type.

in this dataset, we could define matching questions like *"Is this DV suitable for the given dataset?"* and set the answer to be *"Yes"* for positive DVs and *"No"* for negative ones. This kind is refereed as *Type 2* in the following discussion. For the *Data Retrieval* and *DV Structure-related* questions, we employ a rule-based method to generate these questions with answers, which is also a common practice in other QA-related tasks such as [21]. It is noteworthy that this category of questions is among the most challenging in the FeVisQA task, and the model's performance on these questions best reflects its level of understanding of DVs. This kind is refereed as *Type 3* in the following discussion. After all the above-mentioned process, we finally obtained a large-scale FeVisQA dataset with its statistics shown in Fig. 2(b) and Fig. 2(c). It should be noted that similar methodologies used in constructing the dataset are also prevalent in other QA studies such as [21].

III. Our Proposed DNN-based Approach

In this section, we mainly discuss the structure of the proposed FeVisQANet model.

A. Model Overview

After discussing the basics and the dataset, we are ready to introduce the details of our proposed FeVisQANet model. Comprising the DV specifications, the related datasets, and the question as the inputs, it then generates the textual answer corresponding to the given question. As shown in Fig. 3, the whole network is composed of two main components: a multi-modal encoder and an adaptive decoder, the former which extracts comprehensive information from the specification, the question as well as the datasets, and the latter decoding the desired answer according to the type of question. The multi-mode encoder is composed of popular structures like Transformer, taking multi-modal information as inputs and integrating them into a latent representation with the fusion mechanism. The decoder uses the hidden representations from the multi-mode encoder and adaptively generates different answers according to the types of questions. In this section, we would like to present more details about each component.

B. Multi-Modal Encoder

1) Question and Dataset Encoder: The question and dataset are strongly correlated and essential for retrieving or inferring the correct answer. As such, we design a novel architecture to encode the question and table simultaneously, and preserve their information, respectively. An embedding layer, initialized from the well-trained pre-trained model Tapas [22], is employed here. More specifically, we provide the question and the related tables as the inputs to the Tapas tokenizer and then go through an embedding layer to obtain an effective initial embedding Z_q. The main structure following the above module is based on the stacked Transformer encoder blocks, where each block comprises a multi-head attention layer and a feed-forward layer. The multi-head attention works by calculating the relevance score among the input text during each head and using the score to update the hidden states. The essence of each head is a scale-dot attention, denoted as Eq. (1), where Q is the query, K is the key, and V is the key. After that, a new Q, which is fused by the weighed V, is generated as

$$\texttt{attention} = \texttt{Softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{1}$$

where $\frac{1}{\sqrt{d_k}}$ is the scaling factor with a d_k equal to the dimensionality of query Q.

The attention mechanism in the Transformer encoder is self-attention where Q, K, and V refer to the same vector. Specifically, they are linearly projected from the inputs of t-th block $H_q(t)$, and the initial input equals the output of the embedding layer, i.e., $H_q(0) = Z_q$. We obtain the queries, keys, and values through multiplication with the respective trainable weight matrices W_i^Q, W_i^K and W_i^V, so the i-th head inherited from Eq. (1) can be represented as Eq. (2). Then all heads constitute the outputs of the current multi-head attention layer. Afterward, previous outputs go through a feed-forward layer consisting of a linear transformation, ReLU activation, and another linear transformation to get the current block outputs, shown in the following equations. Finally, the hidden representation of the question and table is generated from the last block as H_q.

$$head_i = \texttt{attention}(H_q(t)W_i^Q, H_q(t)W_i^K, H_q(t)W_i^V), \tag{2}$$

$$\texttt{MultiHead}(Q, K, V) = [head_1; \cdots; head_h]W^O, \tag{3}$$

$$H_s = \texttt{FFN}(\texttt{MultiHead}(Q, K, V)) \tag{4}$$

2) DV Specification Encoder: The DV specification is in the form of a JSON object hierarchically representing DV information. It indicates information like the calculation of the datasets, the arrangement of data, and the visualization representation. Thus, it is crucial to infer the answer given a question about the DV's understanding. To capture both the structural and the semantic information involved in a specification, we first convert it into an Abstract Structure Tree (AST) [23] and then use a Transformer-based structure to encode it into some hidden representations.

Representing Specification with AST. AST enjoys the benefits of preserving the same semantics and structure as the JSON object and can also be easily processed by neural networks. To give a better illustration, we present an example in Fig. 4 to illustrate how to convert a JSON object into an AST. More specifically, the root node is initialized first, and then, the tree is constructed in a recursive style. Given a key-value pair in the JSON, the key is set as a parent node in the AST. Then, the value is directly set as a child node when the value is a string or a number (e.g., as for the item 'mark', node 'mark' only has one child node 'bar'). Alternatively, is set as a nested item set of child nodes when the value is a nested dictionary (e.g., for the item 'encoding', its value is a nested dictionary, so all keys in the dictionary, including item 'x' and item 'y', are regarded as the child nodes). We repeat this process until all the branches reach the leaf nodes.

Encoding AST with Transformer. After the procedure of the AST construction, we need to focus on both the parent and the child nodes to get complete and concise information on the DV specification. A Transformer-based encoder is employed here to encode the specification into a hidden representation.

Specifically, after we obtain the AST tree by above conversion, and we conduct a *pre-order traversal* [24] to get

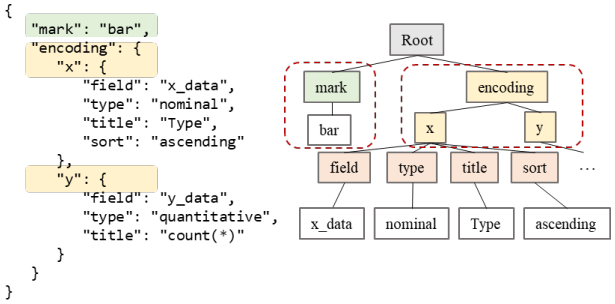


Fig. 4: Example of converting specification to AST

the corresponding sequence. Then, we convert the word in the sequence into embedding by the well-trained Tapas, and construct the representation of the whole sequence by a Transformer. We represent the representation as H_v.

3) Cross-modal Encoder: Once we obtain the question-table embedding H_q and specification embedding H_s, we intend to fuse them while keeping their features. Similar to the Transformer encoder, this cross-modal encoder referring to [25] is composed of several blocks, and each block contains a cross-attention module, a self-attention module, and a feed-forward network module. In particular, there are two branches in each block, one for the question table and one for the specifications. As shown in Fig. 3, taking feature1 and feature2 as input and assigning different values to Q, K, and V, these two branches perform identical operations. Feature1 and feature2 are specified to H_q and H_s respectively, and $Q = H_q$, $K = V = H_s$ for the question-table branch and vice versa in the specification branch. Each block will output a new syncretic representation and go into the next block until the whole network is finished. Finally, we obtain new question-table embedding H_q and specification embedding H_s, which serves for the following decoders.

C. Answer Decoder

The free-form question-answering task indicates that the answer is not in a fixed format, which means that it can be in any length or format. In particular, we summarize existing questions and their corresponding answers into three basic types: long-form string, number, or boolean value. A potential model should give a long explanatory text to answer the questions querying the meaning of a DV, a short boolean value to questions about the quality of a given DV, and a short text or numbers to questions that require data retrieval or logical reasoning about the data and DV. Therefore, we design an adaptive decoder to generate different answers to handle this task and generate more accurate answers. In detail, three sub-generators are designed: a textual generator, a classifier, and a retrieval-based calculator.

1) Textual Generator: The textual generator aims to predict long text as answers. We build our model using a Transformer since it performs very well in generating a long text. The Transformer-based decoder is similar to the encoder described in Section III-B1, which is also based on stacked blocks utilizing multi-head scaled dot attention, except a unique encoder-decoder cross-attention layer is concluded in each block.

More specifically, a self-attention layer takes the previously predicted word embedding as the inputs, which is assigned to Q, K, and V. Then a cross-attention layer takes the outputs from the last self-attention layer as query Q and performs a weighted aggregation on value V, which refers to the cross-modal encoder output. Due to these designs, the correlated text from the encoder can be captured and served for the next word prediction.

2) Classifier: The objective of this decoder is to distinguish whether a description is right or wrong, so a binary classifier is implemented here. In particular, a pooling module, which consists of a dense layer and a *Tanh* activation layer, takes the hidden state of the first token as the input to obtain an exclusive representation for the textual inputs. Then a linear projection followed by a *Sigmoid* activation is further employed to get the final binary outputs.

3) Retrieval-based Calculator: Inspired by Table PLMs like Tapas, we designed a retrieval-based calculator to answer data retrieval and logic reasoning questions. Our decoder contains two classifiers to select the cells and choose the aggregation operator. Then we can infer the final results based on the original cell value or calculated values from the selected cells. The final output from the cross-modal encoder can be divided into two parts, a question embedding, and a table embedding. The table embedding is then used to compute the *logit* value for each token, and an average vector is computed from all the tokens in a cell, which is regarded as the *logit* value for the cell. As for the classifier for aggregation, considering the actual need in our dataset, extra operators such as 'yes', 'no', and 'diff' are added to ensure that this decoder can handle any question produced from the table. The operator is chosen according to the outputs of a linear layer followed by a *softmax* when taking the hidden state of the first token.

D. Model Training

Since all types of questions will share one common encoder and then go through different decoders to predict the answers, a batch is divided into min-batches where different types of questions are categorized into different mini-batches. The decoders are updated when we feed the mini-batches into the model one by one and then the common encoder is updated at the end of this batch. Although the decoders are different, the essence of the final prediction can be regarded as a classification problem. Thus widely used cross-entropy loss is applied here. The details are shown as follows.

(i) As for the textual generator, the target is to maximize the probability of the expected word at each step, as shown in Eq. (5),

$$\mathcal{L}_1(y, \hat{y}) = -\frac{1}{N_T}\sum_{n=1}^{N_T}\frac{1}{n_t}\sum_{t=1}^{n_t}\log p_{out}^t(y = \hat{y}_n^t), \tag{5}$$

where y and $\hat{y}$ are the predicted and target sequence. N_T is the total number of training samples in this mini-batch, and n_t is the length of the n-th target sequence. $p_{out}^t(y = \hat{y}_n^t)$ indicates the probability that the model chooses the target word $\hat{y}_n^t$ to be the final output at step t of the n-th sample.

(ii) When it comes to the binary classifier, the binary cross-entropy loss shown in Eq. (6) is employed

$$\mathcal{L}_2(y, \hat{y}) = -\frac{1}{N_C}\sum_{i=1}^{N_C}(y_i \log p_i + (1 - y_i)\log(1 - p_i)), \tag{6}$$

where N_C is the total number of questions that in this category, and y_i and p_i are the true category and the predicted probability for i-th sample, respectively.

(iii) As for the retrieval-based calculator, we train the model to predict a binary label for each cell and select the most probable aggregation operator. To be more specific, the average binary cross-entropy loss over all cells is used here, as shown in Eq. (7),

$$\mathcal{L}_{cell} = -\frac{1}{N_R}\sum_{n=1}^{N_R}\frac{1}{c_n}\sum_{i=1}^{c_n}(y_i \log p_i + (1 - y_i)(1 - \log(1 - p_i)), \tag{7}$$

where N_R is the total number of question that belongs to this category, and c_n is the number of the cells in the n-th concerned table. The cross-entropy loss over aggregation choice is defined as Eq. (8), where $p_a(y_a = \hat{y}_a^n)$ is the probability when the predicted operator y_a is equal to the target one $\hat{y}_a^n$ for n-th sample. Then, these two loss functions are minimized simultaneously.

$$\mathcal{L}_{agg} = -\frac{1}{N_R}\sum_{n=1}^{N_R}\log p_a(y_a = \hat{y}_a^n), \tag{8}$$

$$\mathcal{L}_3 = \mathcal{L}_{cell} + \mathcal{L}_{agg}, \tag{9}$$

IV. Our Proposed LLM-based Approach

In this section, we will provide a detailed introduction to our proposed LLM-based framework, VisQA.

A. Model Overview

As shown in Figure 5, the VisQA framework is composed of several stages: Question Classification, Initial Answer Generation, RAG-based Reasoning, and Answer Validation and Correction. This framework cleverly employs the classifier of FeVisQANet to categorize questions, determining whether RAG technology is needed to assist in the Initial Answer Generation process and whether a DV chart is needer as an image input. This significantly reduces computational resource consumption during the process. Additionally, with the aid of RAG technology, the framework generates reasoning processes for relevant examples, which are used to validate and, if necessary, correct the initial answer. This framework fully leverages the potential of MLLMs in FeVisQA tasks. In this section, we will provide a detailed explanation of these four steps.

B. Question Classification

In the FeVisQA task, samples necessitating a textual response must leverage RAG technology to aid in formulating answers. This approach simulates response styles in analogous scenarios and enhances performance on metrics such as METEOR. For samples requiring a classification response,

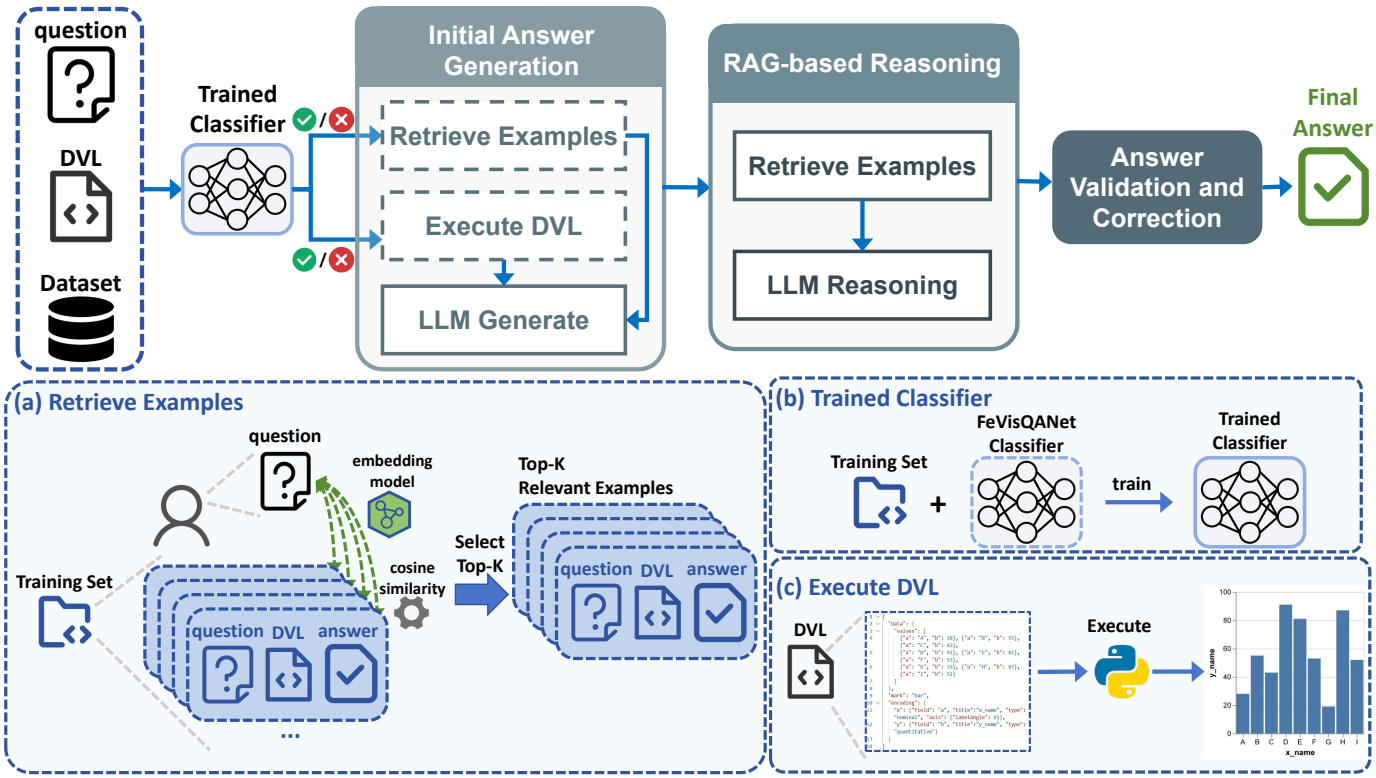


Fig. 5: The working pipeline for our proposed LLM-based approach. For each input (question, DVL, dataset) pair, the following steps are executed: (i) First, classify the input (question, DVL, dataset) pair to determine whether RAG technology assistance and whether the image input assistance are required. (ii) Then, based on the classification result, perform (or skip) the example retrieval process and the DVL execution process, and then generate the initial answer; (iii) Next, retrieve the top-K (question, DVL, answer) pairs and use the LLM to generate the reasoning processes for these examples based on the answers; (iv) Finally, use the reasoning processes of these examples as references to verify and, if necessary, correct the initial answer.

it is imperative to accept images as input. This necessity arises from the challenge of determining whether the final DV chart effectively represents the data solely based on the DV specification (e.g., highly discrete data is not suitable for a bar chart but is better represented by a scatter chart). Conversely, for samples demanding precise numerical calculations, the related examples and image inputs provided by RAG technology serve as distractions, impeding accurate reasoning about the current problem. Consequently, we stipulate that *Type 1* questions require RAG technology assistance, *Type 2* questions necessitate the DV chart image as additional input, and *Type 3* questions require neither RAG technology assistance nor the DV chart as additional input.

To accurately classify the questions, we adopted the Encoder structure from FeVisQANet and employed the Classifier Decoder structure from FeVisQANet as the model's Decoder. Other implementation details remain consistent with FeVisQANet and are also based on the PLM model Tapas.

C. Initial Answer Generation

After classification by the FeVisQANet-based classifier, we need to construct the prompt for Initial Answer Generation based on the classification results, which involves retrieving relevant examples and executing the DV specification. Specifically, during the retrieval of relevant examples, we first use a pre-trained text-to-embedding model to convert the input question and DV specification into embeddings, and then calculate the cosine similarity with the embeddings of examples in the example library. The formula is as follows:

$$\mathtt{sim_q} = 1 - \mathtt{cosine}(\mathtt{q_{in}}, \mathtt{q_{lib}})$$
$$\mathtt{sim_{dvl}} = 1 - \mathtt{cosine}(\mathtt{dvl_{in}}, \mathtt{dvl_{lib}})$$
$$\mathtt{sim} = \mathtt{sim_q} \times 0.5 + \mathtt{sim_{dvl}} \times 0.5$$

where $\mathtt{q_{in}}$ and $\mathtt{dvl_{in}}$ are the vector representations of the input question and DVL, and $\mathtt{q_{lib}}$ and $\mathtt{dvl_{lib}}$ are the vector representations of the questions and DVLs in the example library. The $\mathtt{cosine()}$ is the method to calculate the cosine distance between two vectors. The $\mathtt{sim_q}$ and $\mathtt{sim_{dvl}}$ represent the semantic similarity between the questions and DVLs of the examples, respectively, and sim represents the overall similarity (relevance) between the examples. Finally, the top-K examples with the highest similarity are selected as contextual references.

Due to space limitation, the prompt template for *Type 3* questions is shown in the online technical report provided in the footnote of Section I, which do not rely on RAG technology or image input assistance. Instead, we provide clear instructions for the LLM to perform free-form reasoning.

The prompt templates for *Type 1* and *Type 2* questions are similar to the above prompt template but use concise

TABLE I: Performance Comparison

	Type 1			Type 2	Type 3	Total			
Method	BLEU-4	ROUGE	METEOR	EM	EM	BLEU-1	ROUGE	METEOR	EM
Seq2Seq	0.0505	0.3171	0.3278	**99.96%**	48.86%	**0.5864**	0.5733	0.3151	53.00%
Transformer	0.0671	0.3470	0.3330	**99.96%**	48.05%	0.5830	0.5728	0.3160	52.40%
DualEncoder	0.0585	0.3320	0.3317	99.67%	47.08%	0.5758	0.5629	0.3106	51.62%
Zero-Shot LLM	0.0062	0.1661	0.1949	64.46%	53.71%	0.0086	0.5493	0.2831	50.51%
Few-shot LLM	0.0223	0.2321	**0.3822**	64.09%	51.87%	0.0111	0.5184	0.3087	52.19%
RAG for LLM	0.0398	0.2563	0.3799	69.15%	58.40%	0.0120	0.5781	0.3230	55.12%
Fine-tuned Llama	0.0609	0.3396	0.3715	89.03%	62.94%	0.0119	0.6725	0.3577	61.86%
FeVisQANet (Ours)	**0.0791**	**0.3749**	0.3764	99.63%	73.46%	0.5756	0.6707	0.3393	71.35%
VisQA (Ours)	0.0116	0.2237	0.3151	77.80%	**89.79%**	0.0202	**0.8205**	**0.4496**	**79.93%**

instructions. After the Initial Answer Generation process, we obtain the initial answer.

D. RAG-based Reasoning

To automatically provide a similar and reasonable reasoning process for each example to verify and, if necessary, correct the initial answer, we retrieve the top-K most relevant (question, DVL, answer) pairs based on the input question and DVL. These pairs are then constructed into a prompt, allowing the LLM to refer to the answers and provide a reasoning process. The advantage of this approach is that it enables the automatic generation of possible reasoning processes based on different inputs, rather than using a fixed reasoning process, thereby improving the accuracy of initial answer verification and correction.

E. Answer Validation and Correction

After obtaining the reasoning processes of the relevant examples, we use them as references to perform reasoning on the current (question, DVL) pair, thereby verifying whether the initial answer is correct and correcting it if it is not. This step allows VisQA to reflect on the answers it generated based on the examples in the example library.

V. Experimental Setup

In this section, we provide a detailed performance evaluation of our proposed task and model in terms of quantitative metrics. We first introduce the experimental setup, evaluation measurements, and the designed baselines and then demonstrate the effectiveness of the proposed model by comparing them with these strong baselines.

A. Baselines

FeVisQA has not been studied in the literature, and we need to construct various FeVisQA baselines in our experiment to compare the performance. The detailed descriptions can be found as follows.

- **Seq2Seq**: Seq2Seq [26] is the pioneering study that tackles many NLP tasks (e.g., automatic speech recognition) in an encoder-decoder style. Our implementation converts the FeVisQA problem into a machine translation problem.
- **Transformer**: Transformer [27] has recently been dominating many areas [27]–[32]. It could greatly surpass previous methods. As such, we also employ a Transformer as a baseline for performance comparison.
- **DualEncoder**: DualEncoder is a popular CodeQA model proposed in [16], which improves the basic Seq2Seq model with two encoders.
- **Zero-shot LLM**: The Zero-Shot LLM is an experimental setup designed to guide the LLM in completing the prediction task for FeVisQA by crafting clear and precise instructional prompts, rather than providing examples.
- **Few-shot LLM**: The Few-shot LLM is a key way to use in-context learning (ICL). It adds several examples to the context to help LLMs learn how to handle specific task.
- **RAG for LLM**: RAG [33] technology can smartly pick examples from the knowledge base as references based on the input, helping to reduce model mistakes. The retrieval method is the same as the one described in Section IV-C.
- **Fine-tuned Llama**: The Fine-tuned Llama is an experimental setup obtained by performing `Lora` fine-tuning on `Llama-3.2-1B` using the FeVisQA training dataset, representing the performance of model fine-tuning methods on the FeVisQA task.
- **FeVisQANet**: FeVisQANet is our proposed DNN-based model that includes an advanced multi-modal encoder and decoder designed explicitly for this FeVisQA task.
- **VisQA**: VisQA is our proposed LLM-based multi-step framework. By leveraging RAG technology and a complex framework, it fully harnesses the performance of MLLMs in the FeVisQA task.

We split the FeVisQA into a training set and a testing set. To ensure fairness, all the methods were trained on the same training set and evaluated on the same testing set.

B. Evaluation Metrics

We follow the common metrics in the CodeQA task and also employ these widely used metrics to compare the performance, namely *BLEU* [34], *ROUGE* [35], *METEOR* [36], *EM* [16]. The detailed definitions are as follows.

- **BLEU**: BLEU, short for BiLingual Evaluation Understudy, is the most widely used metric in NLG tasks. In BLEU, the calculation of n-gram precision and recall are modified, which counts the percentage of N-gram co-occurrences between candidate and reference texts. The formal definition of BLEU is:

$$\texttt{BLEU} = \texttt{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), w_n = 1/n \quad (10)$$

where p_n is the modified n-gram precision. The brevity penalty BP is calculated as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}, \tag{11}$$

where c is the length of the candidate text and r is the effective reference text length. In this paper, we reported BLEU-4 for all our experiments.

- **ROUGE**: ROUGE is short for Recall Oriented Understudy for Gisting Evaluation. Based on different features used to compute recall, ROUGE includes types including ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. ROUGE-L is based on the longest common subsequence (LCS), and uses F-score instead of using only recall. We report the ROUGE-L score in our experiment.
- **METEOR**: METEOR stands for Metric for Evaluation for Translation with Explicit Ordering, which is another popular evaluation metric for text generation tasks. Compared with BLEU, METEOR could reflect a better correlation with human judgment. METEOR is formally defined as:

$$\texttt{METEOR} = F_{mean}(1-p), \tag{12}$$

where F_{mean} is the *harmonic mean* of the unigram's precision P and recall R,

$$F_{mean} = \frac{PR}{\alpha P + (1-\alpha)R}, \tag{13}$$

where α is a parameter between 0 and 1 with the default value 0.9. The text segment's penalty p function is defined as

$$p = \gamma(\frac{c}{m})^{\beta}, \ \texttt{where } 0 \leq \gamma \leq 1, \tag{14}$$

where c is the number of chunks and m is the number of mapped unigrams between two texts, β is a parameter with the default value set to be 3.

- **EM**: EM stands for Exact Match; Compared with the above metrics, EM directly count the percentage of the correctly predicted answers. However, it is more suitable for evaluation short answers. EM is formally defined as:

$$\texttt{EM} = \frac{N_e}{N}, \tag{15}$$

where N_e is the number of exact matches, and N is the total number of all the instances.

C. Implementation Details

Our models are trained by the Adam optimizer, with a mini-batch size and a learning rate set to 64 and 1e-4, respectively. We use the pre-trained embedding weight from the Tapas-base model with the hidden size set to 768, which is also used as the hidden size for other encoder and decoder modules. At the same time, the initial embeddings of the DV specifications come from the Glove with dimension set to 300, followed by a linear layer to transfer the dimension to 768. The number of the Transformer layer is set to 2 and the number of heads is set to 4 for all Transformer-based modules and the GCN layer is set to 1. As for the textual generator, the vocabulary size is set to 5614, which is decided by the training corpus. The parameters of the Calculator also follow the same settings from Tapas-base model, except that there exist seven different aggregation operators, namely 'none', 'count', 'sum', 'average', 'yes', 'no', and 'diff'. During the inference phase, we use the selected operation and columns to obtain the final result. In the baseline models, the LLMs used are all "gpt-4o-mini-2024-07-18", which include Zero-shot LLM, Few-shot LLM, RAG for LLM, and VisQA, with the parameter set to "temperature=0.0". For the Fine-tuned Llama, the model used is "Llama-3.2-1B", employing the LoRA fine-tuning method, with lora_dropout and lora_rank set to 0 and 8, respectively.

D. Experimental Results

1) Comparison of Accuracy: Table I shows our models outperform baselines on FeVisQA, especially on challenging Type 3 questions. We demonstrate our models' advantages through performance comparisons.

VisQA achieves state-of-the-art performance, significantly outperforming all baselines, particularly on Type 3 questions. VisQA reaches 89.79% EM on Type 3 and 79.93% overall EM, a substantial improvement over Fine-tuned Llama and FeVisQANet, establishing VisQA as the accuracy leader.

Analyzing by model category reveals prompting-based multi-step LLMs like VisQA are most effective. Traditional DNNs (Seq2Seq, Transformer, DualEncoder) perform weakest. FeVisQANet, a DNN-PLM hybrid, improves over basic DNNs. Fine-tuned Llama performs between FeVisQANet and DNNs. VisQA surpasses all, especially on Type 3, highlighting multi-step LLM prompting's effectiveness.

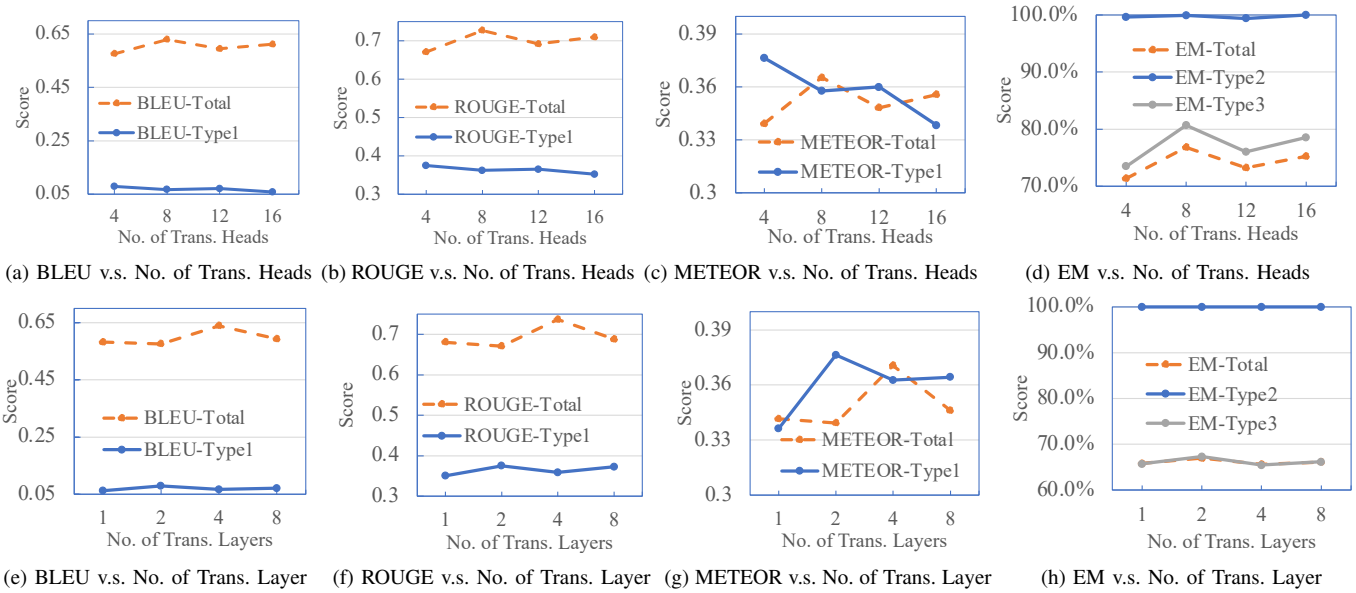
Comparing VisQA to direct prompting methods (Zero-shot LLM, Few-shot LLM, RAG for LLM) validates VisQA's advanced prompting. Direct prompting methods are insufficient. RAG for LLM is better but still lags VisQA. VisQA's multi-step framework achieves much higher accuracy, especially on Type 3, confirming its superior prompting architecture for FeVisQA's complexity.

Seq2Seq, Transformer, and DualEncoder baselines highlight FeVisQANet's specialized DNN architecture effectiveness. FeVisQANet outperforms these basic DNNs and performs better than Fine-tuned Llama in overall metrics (71.35% vs 61.86% EM). This shows FeVisQANet advantages for FeVisQA, even versus fine-tuned smaller LLMs. FeVisQANet's design enables better performance with a much smaller parameter size than LLMs, showing parameter efficiency.

Finally, a key advantage of VisQA is its deployment flexibility and adaptability. Unlike fine-tuned models requiring retraining for new data, VisQA operates effectively without FeVisQA-specific fine-tuning. Its performance can be enhanced in real-world applications by simply updating the external knowledge base used in its RAG component. This knowledge base maintenance allows VisQA to adapt to evolving data visualizations and information without model retraining, offering significant practical benefits in dynamic environments.

TABLE II: Ablation Study Results.

	Type 1			Type 2	Type 3	Total			
Method	BLEU	ROUGE	METEOR	EM	EM	BLEU	ROUGE	METEOR	EM
FeVisQANet	**0.0791**	**0.3749**	**0.3764**	99.63%	**73.46%**	0.5756	**0.6707**	0.3393	**71.35%**
- w/o Dataset	0.0651	0.3599	0.3461	95.23%	28.61%	0.4262	0.4199	0.2352	37.07%
- w/o Adaptive Decoder	0.0678	0.3600	0.3512	**99.96%**	55.34%	**0.6383**	0.6283	**0.3455**	57.85%
Few-shot LLM	0.0223	0.2321	**0.3822**	64.09%	51.87%	0.0111	0.5184	0.3087	52.19%
RAG for LLM	**0.0398**	**0.2563**	0.3799	69.15%	58.40%	0.0120	0.5781	0.3230	55.12%
VisQA	0.0116	0.2237	0.3151	77.80%	**89.79%**	0.0202	**0.8205**	**0.4496**	**79.93%**
- w/o Correct	0.0197	0.2416	0.3421	**82.74%**	87.44%	0.0200	0.8141	0.4476	79.02%
- w Few-shot Generate	0.0191	0.2389	0.3581	77.20%	88.93%	**0.0204**	0.8148	0.4491	79.19%
- w/o Reasoning	0.0076	0.2061	0.2658	60.58%	85.86%	0.0189	0.7619	0.4165	74.11%



(a) BLEU v.s. No. of Trans. Heads (b) ROUGE v.s. No. of Trans. Heads (c) METEOR v.s. No. of Trans. Heads (d) EM v.s. No. of Trans. Heads

(e) BLEU v.s. No. of Trans. Layer (f) ROUGE v.s. No. of Trans. Layer (g) METEOR v.s. No. of Trans. Layer (h) EM v.s. No. of Trans. Layer

Fig. 6: Hyper-parameters Study Result (No. of Trans. Heads and No. of Trans. Layers).

While VisQA leverages powerful LLMs, potentially raising cost or privacy concerns, FeVisQANet remains a valuable alternative: achieving higher accuracy than Fine-tuned Llama with far fewer parameters, making it very strong and efficient, and ideal for resource-limited or privacy-sensitive scenarios.

Due to space limitations in the main text, we have included the error case analysis, the performance comparison between FeVisQANet and VisQA, and the usability study of FeVisQA in the technical report provided in the footnote of Section I.

2) Ablation Studies (The DNN-based Approach): In this section, we conducted ablation studies on the DNN-based approach to demonstrate the contribution of each designed component in FeVisQANet. In particular, we first evaluate the FeVisQANet with all the designed components as the baseline. Then we remove or replace some components of FeVisQANet to check its performance. Each model is then represented by the name(s) of the components it removes or replaces. More precisely, to validate the impact of the dataset on the model's ability to understand data visualization, we removed this part, meaning that the database information was not used as input to the model, and named this experimental setup **w/o Dataset**. For the decoder, we replace the adaptive decoder with a vanilla LSTM-based decoder (**w/o Adaptive Decoder**); The results are shown in Table II.

We take the EM score on the FeVisQA set as the primary indicator, and the other metrics also reflect similar observations. First, incorporating the dataset information brings about a 92.40% relative performance improvement (71.35% v.s 37.07%). This set of ablation studies and the improvement validate the necessity of involving the dataset information in the FeVisQA task, which is usually not included in the existing CodeQA task. This observation validates the necessity of mining and modeling the dataset information in the FeVisQA task. Furthermore, we also observe a 23.34% relative performance improvement using the adaptive decoder.

3) Ablation Studies (The LLM-based Approach): Table II also presents the ablation study results of the LLM-based approach. It highlights the following points: (i) Removing the entire correction process (**w/o Correct**) led to a 0.91% decrease in the overall EM (Exact Match) accuracy and a 2.35% decrease in accuracy for data reasoning (*Type 3*); (ii) Replacing the generate process with the Few-Shot Prompting

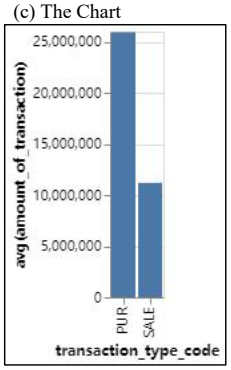
(a) The Dataset

transaction_id	investor_id	transaction_type_code	date_of_transaction	amount_of_transaction	share_count
1	6	SALE	1988/9/16 19:02	302507.7	8718572
2	18	PUR	1982/6/6 17:19	27.257	9
3	2	SALE	1979/4/27 06:03	48777.97	8580
...					
10001	14	PUR	2001/11/28 15:06	4.5263	8040

(b) The Specification


```
{
  "data": {
    "values": [
      {
        "x_data": "PUR",
        "y_data": 25945880.3716125
      },
      ...
    ]
  },
  "mark": "bar",
  "encoding": {
    "x": {
      "field": "x_data",
      "type": "nominal",
      "title": "transaction_type_code",
      "sort": "ascending"
    },
    "y": {
      "field": "y_data",
      "type": "quantitative",
      "title": "avg(amount_of_transaction)"
    }
  }
}
```


(c) The Chart



(d) The QA Predicted by Various Methods

No.	Type	Question	Answer							
			Ground-Truth	Seq2Seq	Transformer	DualEncoder	FeVisQANet	Few-Shot LLM	RAG for LLM	VisQA
1	type 1	what is the meaning of this dv?	create a bar chart showing mean amount of transaction across transaction type code, display by the transaction_type_code in asc.	give me a bar chart to show the average experience working length of journalist working on different role type , show by the y - axis from low to high .	what are the average amount of transaction of transaction , and what is the average amount of amount for each ? , could you show by the name in ascending ?	show the average of amount for different statement detail in a bar chart, and could you show from low to high by the bar ?	show the transaction type code and the average amount of transaction with a bar chart , sort by the x - axis from low to high .	Display the average amount of transactions for each transaction type code in a bar chart.	Show the transaction type codes and the average amount of transactions, and order by the X-axis from low to high.	The DV shows the total amount of transactions for each transaction type code, ordered from low to high on the X-axis.
2	type 2	is this dv a suitable one for the given dataset?	yes	yes	yes	yes	yes	yes	no	yes
3	type 3	what is the difference between the maximum value and minimum value of y-axis?	14747713.13	2449.54	7.388888889	4696.69	14747713.13	14747613.13	14747613.13	14747713.13
4		what is the total number of avg(amount_of_transaction)?	37144047.61	5813.166667	64.72222222	815111.69	37144047.61	37144047.61	37144047.61	37144047.61
5		what is the value of transaction_type_code when avg(amount_of_transaction) is 25945880.3716125 ?	PUR	open	refund	result	PUR	PUR	PUR	PUR

TABLE III: Examples from the FeVisQA dataset with the results predicted by our models and the baselines.

method (**w Few-shot Generate**) resulted in a 0.74% decrease in overall accuracy and a 0.86% decrease in data processing accuracy; (iii) In the experimental setting where the reasoning process that relies on RAG technology for generating related examples was removed and answer correction was directly based on the zero-shot strategy (**w/o Reasoning**), the overall EM decreased by 5.82% and data processing accuracy decreased by 3.93%; (iv) For the Few-Shot LLM and RAG for LLM experimental settings, the overall accuracy (or data processing accuracy) was lower by 27.74% (or 37.92%) and 24.81% (or 31.39%) respectively compared to VisQA. These ablation studies demonstrate the critical roles of all four processes within VisQA, with reasoning being particularly important since it provides a reference reasoning process for verifying and correcting the initial answers, thereby enhancing the performance of the entire framework.

4) Parameter Study: We provide the parameter study to analyze the performance variations concerning the main parameters. We first identify the main factors that affect the performance of FeVisQANet, (i) the number of layers in the Transformer and (ii) the number of heads in the Transformer. The experimental results are shown in Fig. 6. Once again, the primary indicator used here is still the EM score. Fig. 6a $\sim$ Fig. 6d illustrates the first set of results, which discuss how the performance varies concerning the number of heads in the Transformer. When the number of heads exceeds or falls below the ideal value of 8, the performance reaches the its maximum value and then decreases. However, similar phenomena are also observed in the other parameter (e.g., Fig. 6e $\sim$ Fig. 6h), For example, increasing Transformer layers only sometimes results in higher performance (i.e., optimal value 2). This happens because the model's learning capability improves as the layer rises, but using fewer layers does not help because of the smoothing issue with deep neural networks.

5) Case Study: Several cases from the FeVisQA benchmarks are presented in Table III to give a vivid illustration. We include the dataset, the DV specification in Vega-Lite, the chart, the questions, the expected answers, and the predicted answers by various models. In the first example, we have a "Transaction" dataset with many attributes. A potential DV specification [2] is given with various questions. The answers to these questions are diverse. From this example, FeVisQANet and VisQA could accurately understand the meaning of all the questions and then generate the correct answers, whether the question belongs to data retrieval, reasoning, or structure. Other baselines fail to answer most of the questions.

VI. Related Work

In this section, we would like to briefly survey the most related work from these three aspects, DV, (Code) QA, and LLMs and RAG Techniques for DV.

A. Data Visualization

Due to its good visual representation capabilities, various organizations utilize DV to support strategic and operational decisions. Consequently, many DV-related studies have been

[2]We omit some parts of the specification file due to space limits.

conducted in the research and the industry communities [3], [5], [11], [20], [37]–[39]. There are also many Declarative Visualization Languages (DVLs) released in the market, including Vega-Lite [15], ggplot2 [18], ZQL [40], ECharts [19], and VizQL [20]. Among all these DVLs, Vega-Lite is the most common and widely-used one. Two popular tasks have been proposed to lower the barriers to using DVLs to create DV. The first task is *NL2Vis* (i.e., Natural Language to Data Visualization), which allows users to create DVs via natural language descriptions (NLDs). Existing approaches usually tackle this task by converting it into a machine translation and utilizing advanced neural networks. To name a few, Seq2Vis [37] is a representative study that employs a Seq2Seq network to map the NLD into DVs. ncNet [41] is another neural-based model that mainly uses a Transformer with many new optimization techniques like an attention-forcing mechanism. RGVisNet [42] combines the retrieval-based method with the generation-based method that achieves better performance. Besides these neural-based methods, traditional approaches such as the rule-based method are also proposed in studies like [43], [44]. Another prevalent task is *automatic DV recommendation* [45], which focuses on recommending feasible DVs given a massive raw dataset. For example, DataEye [11] proposes a three-step method, namely visualization recognition, ranking, and selection, to tackle this task.

Our proposed FeVisQA task is a new kind of DV-related task that promotes the development of the DV field. FeVisQA enables people to better comprehend DVs and DVLs by logical reasoning when answering questions.

B. (Code) Question Answering

Our work belongs to the General QA (GQA) field, more specifically, CodeQA tasks [16]. GQA has wide applications in the industry. It is usually presented in applications like search engines or dialogue systems. Due to its long history and wide applications, there are a substantial amount of existing studies in the community. A common approach to tackle this problem is first translating the questions into logical forms and then executing the logical forms to get the answers. There are many choices of logic forms, such as lambda-DCS [46], graph query [47], and s-expression [48]. The translation from the question to the logic form could fully use recent advances in the semantic parser, including traditional ones like [46] as well as the neural-based ones [49]. Besides this approach, another commonly used one is to retrieve potential candidates from the knowledge base and then re-rank these candidates to select the top ones as the final results [50]–[53].

When it comes to CodeQA, it is proposed in [16], which focuses on answering questions related to the source code (e.g., Java, Python) for source code comprehension. For example, Liu *et al.* [16] proposed a CodeQA dataset that contains QA pairs on general-purpose programming languages (GPLs), including Java and Python. Then they further designed several methods to tackle this CodeQA task, which is also implemented and compared in our experiments. Later, Lee *et al.* [17] released another CS1QA dataset for educational purposes that include QA pairs with the codes from students and the portion of the code relevant to answering the question.

However, these previous studies only focus on GPLs, and none cover DVLs. Different from GPLs, composing a DV requires defining specifications using DVLs. Moreover, a specification's execution result is a chart, which differs from GPLs. These properties make FeVisQA quite different from models and datasets used for the common CodeQA task. FeVisQA also greatly enlarges and enriches the CodeQA family.

C. LLMs and RAG Techniques for Data Visualization

Retrieval-Augmented Generation (RAG) technology has become a key method to fully harness the potential of LLM in downstream tasks [54], [55]. It has demonstrated outstanding performance in various fields, including but not limited to open-domain QA [56], dialogue systems [57], domain-specific QA [58], and complex tasks such as code generation [59], [60]. By combining information retrieval and generation models, RAG technology not only improves system accuracy but also expands the knowledge range of the models, significantly enhancing their ability to handle various issues in real-world application scenarios.

RAG technology shows progress in DV. Song *et al.* [61] optimized RAG retrieval quality using an Example Mining module for better accuracy.Lu *et al.* [62] addressed performance degradation under input perturbations with a complex multi-step RAG framework. Our VisQA model adopts a similar RAG and multi-step approach to fully harness MLLM capabilities for the FeVisQA task.

VII. Conclusion and Future Work

In this paper, we introduced FeVisQA, a novel Question Answering (QA) task designed to enhance the understanding of Data Visualizations (DVs) by requiring logical reasoning to answer questions about DV specifications for raw datasets. Our primary contributions include: (i) the construction of a benchmark dataset to foster development in this field, and (ii) the proposal of a new multimodal network, FeVisQANet, a novel multi-step framework for MLLMs, VisQA, along with several strong baselines. Extensive experiments demonstrate the feasibility of the proposed task and the effectiveness of our models. We believe FeVisQA can serve as a valuable research benchmark for advancing the domains of data visualization and question answering.

A key limitation of the current FeVisQA benchmark is its diversity, particularly regarding real-world user interactions. Creating a more diverse and representative dataset constitutes a highly promising future research direction.

Acknowledgment

Haodi Zhang and Jinwei Lu are the corresponding authors. We are grateful to the anonymous reviewers for their constructive comments on this paper. Raymond Chi-Wing Wong are supported by fund WEB24EG01-A. Haodi Zhang is supported by the fund 2023B1212010007.

REFERENCES

[1] Y. Song, J. Lu, X. Zhao, R. C.-W. Wong, and H. Zhang, "Demonstration of fevisqa: Free-form question answering over data visualization," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 5417–5420.

[2] M. Aparicio and C. J. Costa, "Data visualization," *Communication design quarterly review*, vol. 3, no. 1, pp. 7–11, 2015.

[3] X. Qin, Y. Luo, N. Tang, and G. Li, "Making data visualization more efficient and effective: a survey," *The VLDB Journal*, vol. 29, no. 1, pp. 93–117, 2020.

[4] S. S. Bhowmick, K. Huang, H. E. Chua, Z. Yuan, B. Choi, and S. Zhou, "Aurora: Data-driven construction of visual graph query interfaces for graph databases," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2689–2692.

[5] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis, "Seedb: Efficient data-driven visualization recommendations to support visual analytics," in *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, vol. 8, no. 13. NIH Public Access, 2015, p. 2182.

[6] D. Ji, H. Luo, Z. Bao, and S. Culpepper, "Navigating data repositories: Utilizing line charts to discover relevant datasets," *Proceedings of the VLDB Endowment*, vol. 17, no. 12, pp. 4289–4292, 2024.

[7] W. Zhang, Y. Wang, Y. Song, V. J. Wei, Y. Tian, Y. Qi, J. H. Chan, R. C.-W. Wong, and H. Yang, "Natural language interfaces for tabular data querying and visualization: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[8] Y. Luo, X. Qin, C. Chai, N. Tang, G. Li, and W. Li, "Steerable self-driving data visualization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 475–490, 2020.

[9] C. Chai, G. Li, J. Fan, and Y. Luo, "Crowdchart: Crowdsourced data extraction from visualization charts," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 11, pp. 3537–3549, 2020.

[10] D. Ji, H. Luo, and Z. Bao, "Visualization recommendation through visual relation learning and visual preference learning," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1860–1873.

[11] Y. Luo, X. Qin, N. Tang, and G. Li, "Deepeye: Towards automatic data visualization," in *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 2018, pp. 101–112.

[12] A. Eldawy, M. F. Mokbel, and C. Jonathan, "Hadoopviz: A mapreduce framework for extensible visualization of big spatial data," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 601–612.

[13] Y. Park, M. Cafarella, and B. Mozafari, "Visualization-aware sampling for very large databases," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 755–766.

[14] M. Krommyda and V. Kantere, "Visualization systems for linked datasets," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1790–1793.

[15] 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, no. 1, pp. 341–350, 2016.

[16] C. Liu and X. Wan, "Codeqa: A question answering dataset for source code comprehension," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 2618–2632.

[17] C. Lee, Y. Seonwoo, and A. Oh, "Cs1qa: A dataset for assisting code-based question answering in an introductory programming course," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022, pp. 2026–2040.

[18] R. A. M. Villanueva and Z. J. Chen, "ggplot2: elegant graphics for data analysis," 2019.

[19] 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," *Visual Informatics*, vol. 2, no. 2, pp. 136–146, 2018.

[20] 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.

[21] K. Kafle, B. Price, S. Cohen, and C. Kanan, "Dvqa: Understanding data visualizations via question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5648–5656.

[22] J. Herzig, P. K. Nowak, T. Mueller, F. Piccinno, and J. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4320–4333.

[23] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 440–450.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[25] A. Masry, X. L. Do, J. Q. Tan, S. Joty, and E. Hoque, "Chartqa: A benchmark for question answering about charts with visual and logical reasoning," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 2263–2279.

[26] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[27] S. M. Lakew, M. Cettolo, and M. Federico, "A comparison of transformer and recurrent neural networks on multilingual neural machine translation," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 641–652.

[28] A. Currey and K. Heafield, "Incorporating source syntax into transformer-based neural machine translation," in *ACL 2019 Fourth Conference on Machine Translation*. Association for Computational Linguistics, 2019, pp. 24–33.

[29] X. Zhao, L. Wang, R. He, T. Yang, J. Chang, and R. Wang, "Multiple knowledge syncretic transformer for natural dialogue generation," in *Proceedings of The Web Conference 2020*, 2020, pp. 752–762.

[30] H. Le, D. Sahoo, N. Chen, and S. Hoi, "Multimodal transformer networks for end-to-end video-grounded dialogue systems," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5612–5623.

[31] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, and H. Ney, "A comparison of transformer and lstm encoder decoder models for asr," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 8–15.

[32] S. Zhou, L. Dong, S. Xu, and B. Xu, "A comparison of modeling units in sequence-to-sequence speech recognition with the transformer on mandarin chinese," in *International Conference on Neural Information Processing*. Springer, 2018, pp. 210–220.

[33] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[34] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[35] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[36] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65–72.

[37] 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*, 2021, pp. 1235–1247.

[38] 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.

[39] N. Tang, E. Wu, and G. Li, "Towards democratizing relational data visualization," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 2025–2030.

[40] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. Parameswaran, "Effortless data exploration with zenvisage: An expressive and interactive visual analytics system," *Proceedings of the VLDB Endowment*, vol. 10, no. 4, 2016.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1533–1544.

[47] Y. Lan and J. Jiang, "Query graph generation for answering multi-hop complex questions from knowledge bases," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 969–974.

[48] Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su, "Beyond iid: three levels of generalization for question answering on knowledge bases," in *Proceedings of the Web Conference 2021*, 2021.

[49] H. Zhang, J. Cai, J. Xu, and J. Wang, "Complex question decomposition for semantic parsing," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4477–4486.

[50] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen, "Open domain question answering using early fusion of knowledge bases and text," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4231–4242.

[51] H. Sun, T. Bedrax-Weiss, and W. Cohen, "Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text," 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)*, 2019, pp. 2380–2390.

[52] W. W. Cohen, H. Sun, R. A. Hofer, and M. Siegler, "Scalable neural methods for reasoning with a symbolic knowledge base," in *International Conference on Learning Representations*, 2019.

[53] J. Shi, S. Cao, L. Hou, J. Li, and H. Zhang, "Transfernet: An effective and transparent framework for multi-hop question answering over relation graph," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 4149–4158.

[54] 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

[55] 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

[56] 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

[57] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen *et al.*, "Check your facts and try again: Improving large language models with external knowledge and automated feedback," *arXiv preprint arXiv:2302.12813*, 2023.

[58] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "Chatlaw: Open-source legal large language model with integrated external knowledge bases," *arXiv preprint arXiv:2306.16092*, 2023.

[59] 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

[60] 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.

[61] 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," *arXiv preprint arXiv:2402.07909*, 2024.

[62] J. Lu, Y. Song, H. Zhang, C. Zhang, and R. C.-W. Wong, "Towards robustness of text-to-visualization translation against lexical and phrasal variability," *arXiv preprint arXiv:2404.07135*, 2024.