Marrying Dialogue Systems with Data Visualization: Interactive Data Visualization Generation from Natural Language Conversations

Yuanfeng Song
AI Group
WeBank Co., Ltd
Shenzhen, China
yfsong@webank.com

Xuefang Zhao
AI Group
WeBank Co., Ltd
Shenzhen, China
summerzhao@webank.com

Raymond Chi-Wing Wong
The Hong Kong University of Science and Technology
Hong Kong, China
raywong@cse.ust.hk

ABSTRACT
Data visualization (DV) has become the prevailing tool in the market due to its effectiveness in illustrating insights in vast amounts of data. To lower the barrier of using DVs, automatic DV tasks, such as natural language question (NLQ) to visualization translation (formerly called text-to-vis), have been investigated in the research community. However, text-to-vis assumes the NLQ to be well-organized and expressed in a single sentence. However, in real-world settings, complex DV is needed through consecutive exchanges between the DV system and the users. In this paper, we propose a new task named CoVis, short for Conversational text-to-Visualization, aiming at constructing DVs through a series of interactions between users and the system. Since it is the task which has not been studied in the literature, we first build a benchmark dataset named Dial-NVBBench, including dialogue sessions with a sequence of queries from a user and responses from the system. The ultimate goal of each dialogue session is to create a suitable DV. However, this process can contain diverse dialogue queries, such as seeking information about the dataset, manipulating parts of the data, and visualizing the data. Then, we propose a multi-modal neural network named MMCVisNet to answer these DV-related queries. In particular, MMCVisNet first fully understands the dialogue context and determines the corresponding responses. Then, it uses adaptive decoders to provide the appropriate replies: (i) a straight-forward text decoder is used to produce general responses, (ii) an SQL-form decoder is applied to synthesize data querying responses, and (iii) a DV-form decoder tries to construct the appropriate DVs. We comparatively evaluate MMCVisNet with other baselines over our proposed benchmark dataset. Experimental results validate that MMCVisNet performs better than existing baselines and achieves a state-of-the-art performance.

CCS CONCEPTS
• Human-centered computing → Visualization; • Computing methodologies → Natural language generation.

KEYWORDS
Conversational Text-to-Visualization, Multi-modal Data Mining

ACM Reference Format:

1 INTRODUCTION
“The future of BI (Business Intelligence) is Conversational! ” – Gartner Analyst Report

Data visualization (DV) has emerged as the prevailing tool in the industry in summarizing the insight behind raw data. Given a massive dataset, DV could vividly and efficiently convey the essence behind the raw data than a verbal description [20]. Due to its great usage, DV has been widely adopted and explored by many commercial vendors (e.g., ThoughtSpot [3] and Microsoft Power BI [2]), as well as academic researchers [13, 33, 43, 44].

To lower the barrier of using DVs, automatic DV tasks, such as text-to-vis (i.e., natural language question (NLQ) to visualization translation), have been extensively explored in the data mining community [30, 31, 43]. For example, Song et al. [43] published a novel text-to-vis model in KDD’22 that employs a hybrid retrieval and generation framework towards accurate DV generation. However, all these existing studies expect the input NLQ to be well-organized and expressed in a single sentence for existing text-to-vis methods. In real-world settings, complex DV is needed through successive exchanges between the DV system and the users. Under this scenario, multi-turn interactions between users and the DV system are frequently required to (1) clarify any ambiguous questions, (2) query the information about the dataset, (3) visualize the data, and (4) notify users about the returned data or charts.

In this paper, we propose a new task named CoVis, short for Conversational text-to-Visualization, aiming to construct DVs via a series of interactions between the system and the user. An illustrative example can be found in Figure 1, which shows that the analyst expresses his data querying and visualization needs gradually when the conversation progresses. The dialogue system generates different responses according to the context, including text, data (by SQL queries) and DV (by Vega-Lite [39], one popular programming language dedicated for DV). The ultimate goal of each dialogue session is to create a suitable DV. However, this process can contain
What is this sales dataset about? This sales table is about the sales records with many attributes like date, price, and amount. The chart is shown as follows:

<table>
<thead>
<tr>
<th>No</th>
<th>date</th>
<th>amount</th>
<th>profit</th>
<th>item_id</th>
<th>discount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1003</td>
<td>2022/1/10</td>
<td>50.03</td>
<td>2.045</td>
<td>2008</td>
<td>0.7</td>
</tr>
<tr>
<td>1004</td>
<td>2022/1/10</td>
<td>38.01</td>
<td>12.04</td>
<td>2007</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Showing the total amount of each day between 2021-07-10 to 2021-07-20, sort by the date from low to high.

Date | Sum(amount)
---|--------
2021-07-10 | 40068.6
2021-07-20 | 56009.0

Select date, sum(amount) from Sales where date >= 2021-07-10 and date <= 2021-07-20 group by date ASC

Show me a line chart for above result

The chart is shown as follows.

Figure 1: A conversation between a user and the data visualization system. The analyst gradually expresses his/her data querying and visualization needs gradually when the conversation progresses. Then, the dialogue system generates different responses according to the context, including text, data, and DV Charts.

The rest of this paper is structured as follows: we first introduce basic concepts and the problem definition in Section 2. Then, we discuss our proposed MMCoVisNet model in Section 3. The experimental results are listed in Section 5, followed by the related work in Section 6. Finally, we conclude the work in Section 7.

2 CONCEPTS AND PROBLEM DEFINITION

This section introduces some foundational concepts that will facilitate a better comprehension of the subsequent work, and these concepts can be divided into two categories, DV and CoVis.

2.1 Concepts on Data Visualization

Declarative Visualization Language (DVL). A DVL specifies the visualization construction details, encompassing chart type, color, size, mapping function, and properties for marks like canvas size and legend. The most common DVLs include Vega-Lite [39], ggplot2 [50], ZQL [41], ECharts [22] and VizQL [16], each of which has its own grammar. Our work mainly uses Vega-Lite due to its popularity and wide usage [30, 31, 43, 46]. However, the methodology proposed can easily be applied to other DVLs.

Visualization Specification. A visualization specification defines the exact properties of a DV using a DVL. In Figure 2, the specification in Vega-Lite is a JSON file specifying properties such as data path, mapping function, and properties for marks like canvas size and legend. The most common DVLs include Vega-Lite [39], ggplot2 [50], ZQL [41], ECharts [22] and VizQL [16], each of which has its own grammar. Our work mainly uses Vega-Lite due to its popularity and wide usage [30, 31, 43, 46]. However, the methodology proposed can easily be applied to other DVLs.

DV Query. To abstract all the DVLs, a inter-media form named DV Query is proposed and widely used in the DV tasks [30, 31, 43, 46].

Figure 2: An example of the visualization specification in Vega-Lite, and its corresponding DV chart.

CoVis could help people better comprehend the data and master how to create suitable DVs. We believe that this new CoVis task will enlarge the family of intelligent dialogue and DV systems.

• To validate the rationale of this new task, we construct a benchmark dataset Dial-NVBench, which could also promote the development of this field. Each dialogue session contains queries like querying the information about the dataset, querying parts of the data, and visualizing the data.

• We design the first multi-modal conversational DV network for this CoVis task - MMCoVisNet. MMCoVisNet is equipped with advanced encoders and adaptive decoders, which guarantees the final inference performance of the whole network.

• We conducted extensive experiments on our proposed dataset and compared it with other popular baselines. The results show that our proposed MMCoVisNet can perform better than several strong counterparts.

More complicated examples, including this one, can be found in the official website of Vega-Lite via https://vega.github.io/vega-lite/examples/.
### Table 1: The Statistics of the proposed Dial-NVBench Dataset

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Train</th>
<th>Val.</th>
<th>Test</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Dialogue Sessions</td>
<td>3115</td>
<td>484</td>
<td>896</td>
<td>4495</td>
</tr>
<tr>
<td>No. of QRs</td>
<td>15044</td>
<td>2359</td>
<td>4332</td>
<td>21725</td>
</tr>
<tr>
<td>Avg. No. of QRs per Session</td>
<td>4.83</td>
<td>4.874</td>
<td>4.824</td>
<td>4.833</td>
</tr>
<tr>
<td>No. of Datasets</td>
<td>137</td>
<td>69</td>
<td>107</td>
<td>143</td>
</tr>
<tr>
<td>No. of General Queries</td>
<td>6197</td>
<td>977</td>
<td>896</td>
<td>4495</td>
</tr>
<tr>
<td>No. of Data-related Queries</td>
<td>5732</td>
<td>896</td>
<td>1229</td>
<td>8259</td>
</tr>
<tr>
<td>No. of DV-related Queries</td>
<td>3115</td>
<td>484</td>
<td>896</td>
<td>4495</td>
</tr>
</tbody>
</table>

43, 46]. Existing text-to-vis studies first synthesizes the DV query from the NLQ, and then executes the DV query to obtain the VegaLite specification. DV query enjoys a SQL-like grammar with visualization component and data manipulation component. The final chart is obtained by rendering the specification file via the visualization engine [39].

### 2.2 CoVis Problem Definition

The definition of the CoVis (Conversational Text-to-Visualization) task is relatively straightforward once the previous concepts are defined. Specifically, given a dataset D, a query q from the user with its context S, the CoVis task aims to predict the response r automatically. CoVis enables users to create suitable DVs interactively. To fulfill the above objective, the context S includes all the previous queries and responses in the same session and they are usually in the DV domain. The query q can be a diverse dialogue query, like seeking the information about the dataset, manipulating parts of the data, and visualizing the data. A desired CoVis model could be represented as \( f(D, S, q) \rightarrow r \). Subsequently, the whole training set can be formally defined as \( \mathcal{T} = \{ (D^{(i)}, S^{(i)}, q^{(i)}, r^{(i)}) \}_{i=1}^{N} \), where \( N \) is the dataset size.

### 2.3 CoVis Dataset Construction

Since CoVis is a new kind of DV-related dialogue tasks, there is currently no literature or public datasets for it. Thus, we first construct and release a benchmark dataset to promote the development of this field. In particular, we synthesize the CoVis dataset by piggy-backing the NVBench dataset [30] from the correlated text-to-vis task. Each sample in the NVBench contains a single-turn NLQ with the corresponding DV (in terms of a DV query). We convert each of them into a dialogue session by (i) randomly deciding the number of rounds (or query-responses (QRs)) for the dialogue; (ii) sampling from some pre-defined general queries about the statistics of the dataset; (iii) generating some SQL-related queries about data manipulation, including ones involving complicated data operations like nested queries and joins, with the method similar to [56]; and (iv) ending each session with a query (the same as the one contained in NVBench) about a DV. To validate this CoVis task, in this new dataset, we mainly focus on data querying and visualization on single tables. Finally, we obtained a dialogue dataset dedicated to DV, and we named it Dial-NVBench (“Dial” is short for “Dialogue”). The statistics of the dataset is shown in Table 1.

### 3 OUR PROPOSED APPROACH

In this section, we mainly discuss the proposed MMCoVisNet model from its main components.

#### 3.1 Model Overview

As shown in Figure 3, MMCoVisNet mainly comprises four components, namely a Dialogue Session Encoder, a Textual Decoder, a SQL decoder, and a DV Decoder. The dialogue session encoder takes the dataset, dialogue history and current query as input and then maps them into some hidden representation. The dialogue history is essential in the CoVis scenario since it helps identify item references in multi-turn dialogue sessions. If the desired response belongs to the general textual category, the textual decoder is used to generate a typical textual response, similar to existing dialogue systems. However, if the query is about manipulating the data, then the SQL decoder is triggered to generate SQL queries to obtain the data. To increase the conversion accuracy, MMCoVisNet employs SemQL [15] as an inter-media form for each SQL query. Similarly, MMCoVisNet uses a DV decoder to obtain the final charts to answer DV-related queries. Following the common practice in text-to-vis

Figure 3: The network structure of the MMCoVisNet model, which is comprised of (i) a Data-aware Dialogue Session Encoder, (ii) a Textual Decoder, (iii) a SQL-form Decoder, and (iv) a DV-form Decoder. To answer the data-related query, MMCoVisNet first generates SemQL via the SQL-form Decoder, then converts the SemQL into an SQL query, and finally obtains the data by executing the SQL query. Similarly, to answer the DV-related queries, MMCoVisNet first generates DV query via the DV-form Decoder, then converts the DV queries to get the Vega-Lite specifications, and finally renders the specifications to get the charts.
\[ s_{d} = d_{1}|d_{2}|\cdots|d_{n}, \]

where \( d_{i} \) is the sequence of the \( i \)-th dialogue session and \( n \) is the total number of the dialogue sessions. Since the dataset information dramatically affects the desired response, we must also preserve the schema information. Moreover, we sample some data records from the dataset as a snapshot for this dataset. To restrict the input sequence length, a certain number of rows are first selected randomly, and after we retain all the names of the schema items, we add the table values from the selected rows word by word until we reach the limit. Finally, the dataset sequence is formatted as

\[ s_{t} = c_{1}|c_{i_{1}}|c_{i_{2}}|\cdots|c_{i_{j}}|\cdots, \]

where \( c_{i} \) is the name of the column \( i \), \( c_{i_{j}} \) is the \( j \)-th value of column \( c_{i} \), and \(' ' \) and \(' \) are the separate characters for the items and rows, respectively.

The input sequence \( Q \) of this session encoder is composed of the current query \( q \), the history sequence \( s_{d} \), and the dataset sequence \( s_{t} \). We first map each token in \( Q \) into its embedding (all these embeddings forms \( Q \)) by a pre-trained Glove [36] and then a bidirectional LSTM network (BiLSTM) [60] is employed to deal with the input embedding to obtain the hidden vectors for the current input. The whole process is represented as

\[ h^{Q}_{t} = \text{BiLSTM}(Q_{t}, h^{Q}_{t-1}), \]

where \( Q_{t} \) is the item from \( Q \) at time step \( t \) and \( h^{Q}_{t} \) represents the hidden states at time step \( t \) for BiLSTM. Finally, we obtain the output \( H_{Q} = \{h^{Q}_{0}, h^{Q}_{1}, \ldots, h^{Q}_{l}\} \) from the session encoder, where \( l \) is the number of tokens in sequence \( Q \).

### 3.3 Adaptive Response Decoder

Since the output of the CoVis task is multi-modal, we design an adaptive decoder to generate diverse responses. The adaptive decoder first uses a classifier to judge the desired response type. Then, three specific decoders are proposed for the downstream process: (i) a textual one for the textual output, (ii) an SQL-form one for the data query, and (iii) a DV-form one for the visualization query.

#### 3.3.1 General Textual Decoder

The objective of this decoder is to generate textual sequences to answer questions about basic information. A Transformer-based decoder that takes the advantage of a multi-head scaled dot-product attention [48] is employed here, where multi-head attention can be formulated as the following equation. Specifically, given the query \( \hat{Q} \), the key \( \hat{K} \), and the value \( \hat{V} \), they are first injected into different high-dimensional spaces

\[ \hat{V} \text{ is the attention mechanism is different from the previously defined } Q \text{ in CoVis, and thus, we use two different variables.} \]
The grammar of SemSQL is represented in Figure 4, where each row in the figure corresponds to an action, and the process of generating SemSQL (which could be represented by an Abstract Structure Tree (AST) \( \hat{r} \)) is achieved by selecting a series of actions.

In particular, each step in the action selection process can be regarded as a classification problem, where the candidates and the final choice are determined by the inputs and outputs of the previous step, represented mathematically as follows:

\[
p(\hat{r}|q, D, S) = \prod_{i=1}^{T} p(a_i|D, S, q, a_{<i}),
\]

where \( a_i \) refers to an action performed at step \( i \), \( a_{<i} \) includes all the previous actions of step \( i \), and \( T \) indicates the total number of actions needed to obtain the desired SemSQL \( \hat{r} \). The actions are further divided into two types: ApplyRule and SelectItem. The former generates the sketch of the AST, the latter selects column and table items to finalize the sketch, and finally, we obtain a complete SemSQL AST. The mathematical details of ApplyRule can be denoted as follows:

\[
h_i = \text{LSTM}([a_{i-1} \oplus e_{i-1} \oplus a_{i-1}], h_{i-1}),
\]

\[
v_i = \text{Softmax}(h_i^T W_h \tilde{H}_Q) \tilde{H}_Q,
\]

\[
u_i = \tanh(W_v[h_i \oplus v_i] + b_v),
\]

\[
p(\hat{r} = a_i|q, D, S, a_{<i}) = \text{Softmax}(\tanh(W_p u_i + b_p)),
\]

where \( \oplus \) denotes the concatenation operation, \( h_i \) is the LSTM hidden state of the current step \( i \), \( a_{i-1} \) is the previous action embedding, \( e_{i-1} \) is the previous action type (i.e., ApplyRule or SelectItem) embedding and \( v_{i-1} \) is the previous contextual representation of LSTM. \( W_h \in \mathbb{R}^{d_m \times d_m}, W_v \in \mathbb{R}^{d_m \times d_m} \) and \( W_p \in \mathbb{R}^{n_a \times d_m} \) (\( n_a \) is the number of actions associated with the specified grammar) are the trainable weights, \( b_v \in \mathbb{R}^{d_m} \) and \( b_p \in \mathbb{R}^{n_a} \) are trainable biases, and the initial state \( h_0 \) is calculated by an average-pooling operation on the output \( H_Q \) given by the encoder (in Section 3.2).

Given the sketch, what we need to do next is to select column items. We first calculate an NLQ-aware representation for column \( c_k \), and then a pointer network is used to select the probability. The mathematical representation of this process is shown as follows:

\[
\hat{p}_{k,j} = \frac{s_{c_k}^T q_j}{\|s_{c_k}\| \|q_j\|},
\]

\[
\hat{s}_{c_k} = s_{c_k} + \sum_{j=1}^{Q} \hat{p}_{k,j} h_j^Q,
\]

\[
Y_{k,i} = (\hat{s}_{c_k})^T W_c u_i,
\]

\[
p(\hat{r} = \text{SelectColumn}(c_k)|D, S, q, a_{<i}) = \frac{\exp(y_{k,i})}{\sum_{i=1}^{Q} \exp(y_{k,i})},
\]

where \( s_{c_k} \) is the embedding for column \( c_k \) obtained from Glove embedding \([36]\), \( q_j \) and \( h_j^Q \) are the corresponding vectors of the \( j \)-th question token extracted from \( Q \) and \( H_Q \), respectively, and \( W_c \in \mathbb{R}^{d_{c} \times d_m} \) is the learnable weights.

### 3.3.3 DV-form Decoder
The ultimate goal of the DV-form decoder aims to show the desired DV chart to the user. Rather than directly generating the charts with the neural network, we design the decoder first to synthesize the DV query. The grammar of the DV query is quite similar to the SQL query, and it was proposed and widely used in the text-to-vis task \([30, 43]\). We use \( r_{sql} \) to represent the SQL query predicted in the last turn by the MMCoVisNet model and \( r_{dvq} \) to denote the DV query predicted by the MMCoVisNet model. If the model synthesizes an accurate DV query \( r_{dvq} \), it yields an accurate DV chart, denoted as \( c \). Specifically, after the DV query is obtained, a pipeline approach shown in Algorithm 1 is triggered to generate the final chart. A DV query is composed of the visualization component and the data manipulation component. The visualization component of the DV query is generated by a Transformer-based decoder in MMCoVisNet, with a shared structure to the textual decoder described in Section 3.3.1. Combined with the previous SQL response and the data retrieved by the SQL query, we can synthesize the visualization specifications \( c_{dv} \) in Vega-Lite.

### 3.4 Model Training
Since there exists different types of responses in one continuous multi-turn dialogue and a similar natural language question, we make sure that they share one session encoder but have different decoders. To train them simultaneously, different QR pairs appear in the same batch and are divided into mini-batches by the question classifier. The mini-batch of various types is fed into the network, and the decoder parameters are updated sequentially (textual-form decoder, data-form decoder and DV-form decoder). Then, the encoder is updated only once in the final phase. The widely-used cross-entropy loss is used in our experiments. For the textual response (\( L_1 \)) and DV-form response (\( L_3 \)), we maximize the probability of each word appearing in the target sequence by the following function:

\[
L_3(\hat{r}, \hat{r}) = L_1(\hat{r}, \hat{r}) = -\frac{1}{N_{i=1}^N} \sum_{i=1}^{N} \sum_{j=1}^{N_i} \log p(r = \hat{r}_{ij}),
\]

where \( \hat{r} \) is the predicted response, \( r \) is the target response. \( N \) is the total number of training samples, and \( N_i \) is the length of the response of the \( i \)-th sample. \( p(r = \hat{r}_{ij}) \) represents the probability of the predicted \( j \)-th token in the \( i \)-th sample being equal to \( \hat{r}_{ij} \).

For the SQL-form response, unlike above mentioned textual and DV-form responses, it works in a two-step generation style. To

<table>
<thead>
<tr>
<th>ALGORITHM 1: The Pipeline for Obtaining the DV Charts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Current query ( q ), Dialogue Context ( S ), MMCoVisNet Model M, SQL Response in the Last Turn ( r_{sql} ), Dataset ( D )</td>
</tr>
<tr>
<td><strong>Output:</strong> Visualization Chart ( c )</td>
</tr>
<tr>
<td><strong>begin</strong></td>
</tr>
<tr>
<td>1. Obtain the DV Query from the model output ( r_{dvq} \leftarrow M(q, D, S) );</td>
</tr>
<tr>
<td>2. Execute the SQL query to retrieval data ( d \leftarrow \text{execute}(D, r_{sql}) );</td>
</tr>
<tr>
<td>3. Get Vega-Lite specification ( c_{dv} \leftarrow \text{transform}(r_{dvq}, d) );</td>
</tr>
<tr>
<td><strong>// Following previous studies like ([30, 43]).</strong></td>
</tr>
<tr>
<td>4. DV Chart ( c \leftarrow \text{render}(c_{dv}) );</td>
</tr>
<tr>
<td><strong>return</strong></td>
</tr>
</tbody>
</table>

| where | "c" |
| "s" |
| "where" |
| "⊕" |
| "𝑎\(\text{AST}\)" |
| "embedding\[36\]," |
| "biases, and the initial state \( h_0 \) are the trainable weights," |
| "rinaing," |
| "items to finalize the sketch, and finally, we obtain a complete SemSQL AST." |
| "where \( a_i \) refers to an action performed at step \( i \), \( a_{<i} \) includes all the previous actions of step \( i \), and \( T \) indicates the total number of actions needed to obtain the desired SemSQL \( \hat{r} \). The actions are further divided into two types: ApplyRule and SelectItem. The former generates the sketch of the AST, the latter selects column and table items to finalize the sketch, and finally, we obtain a complete SemSQL AST. The mathematical details of ApplyRule can be denoted as follows:" |
| "where \( \oplus \) denotes the concatenation operation, \( h_i \) is the LSTM hidden state of the current step \( i \), \( a_{i-1} \) is the previous action embedding, \( e_{i-1} \) is the previous action type (i.e., ApplyRule or SelectItem) embedding and \( v_{i-1} \) is the previous contextual representation of LSTM. \( W_h \in \mathbb{R}^{d_m \times d_m}, W_v \in \mathbb{R}^{d_m \times d_m} \) and \( W_p \in \mathbb{R}^{n_a \times d_m} \) (\( n_a \) is the number of actions associated with the specified grammar) are the trainable weights, \( b_v \in \mathbb{R}^{d_m} \) and \( b_p \in \mathbb{R}^{n_a} \) are trainable biases, and the initial state \( h_0 \) is calculated by an average-pooling operation on the output \( H_Q \) given by the encoder (in Section 3.2)." |
| "Given the sketch, what we need to do next is to select column items. We first calculate an NLQ-aware representation for column \( c_k \), and then a pointer network is used to select the probability. The mathematical representation of this process is shown as follows:" |
| "where \( s_{c_k} \) is the embedding for column \( c_k \) obtained from Glove embedding \([36]\), \( q_j \) and \( h_j^Q \) are the corresponding vectors of the \( j \)-th question token extracted from \( Q \) and \( H_Q \), respectively, and \( W_c \in \mathbb{R}^{d_{c} \times d_m} \) is the learnable weights." |
train the model, we employ the objective of maximizing the log-
likelihood of the ground truth action sequences in the training data,
which is defined as follows:

\[
L_2(r, \hat{r}) = \max \sum_{(D,S, r, q, a^{<i} ) \in \mathcal{T}} \left( \sum_{a_i \in \text{ApplyRule}} \log p(r_i = a_i | D, S, q, a^{<i}) 
\right) + \sum_{a_i \in \text{SelectColumn}} \log p(r_i = a_i | D, S, q, a^{<i})
\]

(17)

4 EXPERIMENTAL SETUP
This section aims to provide a detailed performance evaluation of
the proposed model. We first describe the experimental setup, the
evaluation measurements, and the baselines. Then, we compare the
performance of our proposed model with these strong baselines.

4.1 Models
Several popular dialogue models and methods were utilized to
facilitate a comprehensive performance comparison.

- Seq2Seq: Seq2Seq [7] is a fundamental model that treats the
dialogue task as a translation that maps from the queries to
the responses.
- Transformer: Transformer [48] has shown a dominant per-
formance in many NLP tasks including machine translation
[12], dialogue system [63] and speech recognition [61]. Like
Seq2Seq, we modify it as a baseline method.
- HRED: HRED [40] is a prevalent and representative model
in the field of multi-turn dialog systems.
- MMCoVisNet: This is our proposed method, which employs
a data-aware session encoder as well as advanced adaptive
decoders to generate responses in various forms.

To ensure a fair evaluation, all the aforementioned methods were
trained on the same training set and subsequently assessed on the
same testing set. We try our best to optimize the performance of
these models to achieve their best performance.

4.2 Evaluation Metrics
Since our task includes three kinds of outputs, i.e., the textual form
response, the SQL-form data response and the DV chart response,
we employ various metrics to compare the performance. For the
textual response, we employ the classical BLEU [35], ROUGE [27]
and METEOR [9] scores in a textual-form dialogue system as our
main metrics. For SQL-form response, we employ the widely-used
Sketch Accuracy and Query Matching Accuracy [58] in the text-
to-SQL task as the main indicators. If a model can generate an
accurate SQL query, it will also retrieve the correct data. Similarly,
we also employ Vis Accuracy, Axis Accuracy, Data Accuracy
and DV Accuracy used in text-to-vis [30] to reflect the performance
of the generated DV query. For all these metrics, including the
ones for evaluating the textual, the SQL-form and the DV-form
responses, a larger value of each metric means a better result. A
correctly predicted DV query leads to a correctly generated Vega-
Lite configuration and the final chart.

4.3 Implementation Details
Our networks are constructed using the Adam optimizer, with a
mini-batch size set to 32 for dialogues and a learning rate set to
1 × 10^{-4}. The dimensionality of word embedding of Glove is 300,
the size of vocabulary used in this experiment is 15,894, and each
unseen word is initialized by ‘<unk>’ denoting unknown word. For
the session encoder, we set the number of layers to 1, the number
of heads to 4, and the hidden size to 512. Both textual and DV-form
decoders constructed by the Transformer have the same parameters
as the encoder. In addition, the dimensionality of action embedding
and type embedding in the SQL-form decoder are all set to 128 and
the hidden size to 512.

5 EXPERIMENTAL RESULTS
5.1 Performance Evaluation
We compare the performance in the three forms of the response
generated in Table 2. Since all the baseline methods directly gen-
erate the SQL query rather than first predicting a sketch, only our
proposed MMCoVisNet contains the value in sketch accuracy. From
the result, we report the following observations.

Since the data-related queries are critical and significantly affect
the accuracy of the following generated DV charts, we first evaluate
the performance of SQL-form response generation. The vanilla end-
to-end model, Seq2Seq, performs poorly and is not quite competitive
as a baseline, due to two factors: (i) its capability is limited in
capturing the semantics contained in the dialogue session and the
dataset; (ii) its ability is also limited in generating the SQL queries,
since SQL enjoys a very strict and complicated grammar. Other
advanced models, such as Transformer and HRED, could improve
the vanilla Seq2Seq model from these two aspects. For example, the
HRED model employs a hierarchical decoder to generate responses
for different inputs, leading to a 32.37% relative improvement in
the DV accuracy. The RGvisNet achieves superior performance
on this task owing to its advanced retrieval-generation design,
but it is only capable of producing the final DV output, without
response for the intermediate steps. Prompted LLMs exhibit limited
performance on this task, as they typically work in few-shot in-
context learning style and they are constructed on datasets with
limited DV examples.

Our designed MMCoVisNet could make full use of all the infor-
mation (i.e., dialogue history, query and dataset) out of all possible
models, and at the same time, employ adaptive decoders to generate
accurate responses. Thus, it remarkably beats all the baseline meth-
ods, including advanced models such as HRED and Transformer,
thus proving the efficacy and validating the necessity of designing
an advanced encoder and decoder in the CoVis scenario.

For the final DV-form responses, the basic end-to-end approach,
Seq2Seq, performs poorly and is not quite competitive as a baseline.
Other models (e.g., Transformer and HRED) could beat the vanilla
Seq2Seq model. Even though our proposed MMCoVisNet model
is slightly weaker in the metric Vis. Accuracy (≈ 0.1%), it achieves
the best performance over all other metrics. Most importantly, it
largely surpasses all other baselines in the final DV Accuracy (e.g., a
21.8% relative improvement over the Transformer). We also notice
that the accuracy of the DV-form response is much higher than
the SQL-form query. The reason is also apparent. The proposed
Dial-NBVench dataset contains more SQL-related queries than the DV-related queries, and a large portion of these SQL-related queries are complicated queries (e.g., nested operations), resulting in an overall 62.37% SQL accuracy by MMCoVisNet. Among all these SQL queries, only parts of them are being derived into the final DV and this part is relatively easier, leading to a higher SQL query accuracy compared with the overall SQL query accuracy and thus a higher DV accuracy (i.e., 67.97%).

Finally, we also analyze the quality of the textual form responses generated by various methods. Since this dataset is mainly for DV generation, the textual responses generation part in this dataset mainly focuses on talking about the dataset and not very diverse. And thus, all these models could achieve relative good results. For some metric like BLEU, the difference between these methods is slight (e.g., < 0.76% between Seq2Seq, Transformer and MMCoVisNet). From the result, we could see that our model could still generate comparable results even if baselines such as HRED are dedicated to and specially designed for textual dialogue generation. The main reason for this is that our model can understand the dialogue context, which leads to accurate textual responses.

5.2 Ablation Studies

This section presents ablation studies to show the efficacy and contribution of each designed component within the MMCoVisNet architecture. Firstly, we examine the performance of the MMCoVisNet model with all its proposed components. Then, we investigate the performance of the MMCoVisNet model by removing or replacing its components. We first check each part in the encoder. To evaluate the efficacy of the dialogue session component, we removed it and named this baseline w/o session. Similarly, to analyze the effectiveness of the dataset, we removed it and called it the w/o dataset. We also remove both the dialogue session and the dataset components, and we named it w/o both. For the decoder part, we replace our designed adaptive decoder with a vanilla LSTM-based one (w. basic decoder). The results are shown in Table 3.

We take the DV accuracy as the primary indicator, and the other metrics reflect similar conclusions. First, integrating the session information leads to 51.45% absolute performance improvement (67.97% v.s 16.52%). This ablation study set and its substantial improvement demonstrate the necessity of incorporating dialogue session information in this CoVis task. Other components, like involving dataset information, show similar observations. Then, for the decoder part, using a basic LSTM-based decoder directly generated the SQL query rather than a sketch first, and it has no value in sketch accuracy. Compared with a basic LSTM-based model, our designed adaptive decoder shows about 16.37% relative improvement, proving the necessity of using advanced NN-based models in generating diverse responses.

![Figure 5: Vary No. of Layers in Encoder](image)

![Figure 6: Vary No. of Transformer Layers](image)

5.3 Hyper-parameter Study

This set of experiments analyze the influence of parameter variations on the performance of our proposed MMCoVisNet model. We first identify three main parameters that affect the performance of the model, namely the number of layers in the encoder (Figure 5), the number of heads (Figure 6), and the number of layers in the
Table 4: Examples of Queries & Responses Returned by Various CoVis Methods (errors are marked with red color and ×).

<table>
<thead>
<tr>
<th>Turn</th>
<th>Query</th>
<th>Ground-Truth</th>
<th>CoVis</th>
<th>Seq2Seq</th>
<th>Transformer</th>
<th>HRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>give me a short description about the table</td>
<td>a product table with many attribute like product id, product name, product price, product description, other product service detail</td>
<td>a product table with many attribute like product id, product name, product price, product description, other product service detail</td>
<td>a product table with many attribute like product id, product name, product price, product description, other product service detail</td>
<td>a product table with many attribute like product id, product name, product price, product description, other product service detail</td>
<td>a product table with many attribute like product id, product name, product price, product description, other product service detail</td>
</tr>
<tr>
<td>2</td>
<td>What are the different product names and the average product price for each of them</td>
<td>SELECT Product_Name, avg(Product_Price) FROM PRODUCTS GROUP BY Product_Name</td>
<td>SELECT Product_Name, avg(Product_Price) FROM PRODUCTS GROUP BY Product_Name</td>
<td>select product_id , avg (product_price) from product group by product_name</td>
<td>select product_name, avg (product_price) from product group by product_name</td>
<td>select product_description , avg (product_price) from product group by product_name</td>
</tr>
<tr>
<td>3</td>
<td>how about the minimum product price</td>
<td>SELECT Product_Name , min(Product_Price) FROM PRODUCTS GROUP BY Product_Name</td>
<td>SELECT Product_Name , min(Product_Price) FROM PRODUCTS GROUP BY Product_Name</td>
<td>select product_id , min (product_price) from product group by product_name</td>
<td>select product_price , min (product_price) from product group by product_price</td>
<td>select product_name , min (product_price) from product group by product_name</td>
</tr>
<tr>
<td>4</td>
<td>show me a bar for above result</td>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
<td><img src="image3.png" alt="Graph" /></td>
<td><img src="image4.png" alt="Graph" /></td>
<td><img src="image5.png" alt="Graph" /></td>
</tr>
</tbody>
</table>

(a) Textual-form Perf. (b) SQL-form Perf. (c) DV-form Perf.

Figure 7: Vary No. of Transformer Heads

Transformer (Figure 7). We change the value of one parameter and fix the remaining ones each time.

We utilize the DV accuracy in Figure 5c as the main indicator in the following analysis. The highest accuracies are obtained when the encoder comprises three layers, as the performance declines when the number of layers deviates from this optimal value, whether higher or lower. In particular, we observe that increasing this value does not always lead to a better performance. This phenomenon is also observed in previous studies like [24], since too many layers usually result in the smoothing problem. For other parameters, i.e., the number of Transformer layers (Figure 6) and the number of Transformer heads (Figure 7), we also observe similar performance trends.

### 5.4 Case Study

We also list a real case shown in Table 4 to give a vivid illustration, which includes the original datasets to be analyzed, the queries and the predicted responses (text, SQL query & data, and DV charts) by various methods. In this example, the user directly queries the details about a “Product” dataset by asking general question like “give me a short description about the table”, data-related question like “what are the different product names and the average product price for each of them”, and DV-related question like “show me a bar for the above result”.

From this example, we could see that MMCoVisNet could accurately understand the meaning of all the queries and generate correct answers for text, data and DV. Other baselines, including the advanced Transformer and HRED, fail in answering most of the queries and also obtained incorrect DV charts finally. It is noted that MMCoVisNet also makes full use of the meaning in the whole dialogue session rather than a solely current query, which is very important for understanding queries like “how about the minimum product price” (Turn 3) that refers to the previous context (Turn 2).

### 6 RELATED WORK

This work is relevant to the research areas of data visualization, natural languages to SQL, and dialogue systems.

#### 6.1 Data Visualization

Recent years have witnessed an enormous rise in DV in the data mining [14, 18, 37, 43] and the database communities [16, 29, 30, 46, 46].
47]. There are many intelligent tasks proposed to lower the barriers to use DV, including text-to-vis and DV recommendation. Text-to-vis focuses on automatically translating a natural language question (NLQ) into its corresponding DV. It also enables non-experts to manipulate the DV system easily. The most popular method right now is to treat it as a machine translation problem and then utilize deep neural networks to map from the input (i.e., NLQ) to the output (i.e., DV). For example, Cui et al. introduces text-to-viz and utilizes rule-based methods to translate text statements to infographics [11]. Draco-Learn [32] views the visualization design as a selection process from a set of constraints and then use experimental data to turn the weights for these soft constraints. Data2Vis [13] also attempts to convert the data series to visualization specifications in some declarative languages in a machine translation style. NL4DV [33] is a Python toolkit that offers a number of high-level operations to assist users in building DV systems with NL-interfaces (NLIs). To address the challenge of limited dataset in this field, Luo et al. additionally describes an approach to synthesize the NLQ-DV dataset called NVBench based on the popular NL2SQL benchmark Spider [58]. A Seq2Seq model is further trained on this dataset to validate the feasibility of DV query generation from NLQs on this benchmark [36]. RGVisNet [43] is another representative study in KDD’22 that employs a DNN-based approach for converting NLQs into DVs. However, it beats other methods by combining the retrieval-based method with the generative-based one. In contrast, automatic DV recommendation directly outputs top-N potential DVs given a dataset without any NL queries from the users. To name a few, DataEye [28] tackles the DV recommendation problem in a three-step style, namely visualization recognition, ranking and selection. Qian et al. [37] employ an end-to-end learning-based method to construct DVs, giving a massive dataset.

Different from these studies, we propose a novel DV-related task named CoVis (i.e., Conversational text-to-Vis), which focuses on interactive exchange between the user and the DV system. With the benchmark and all the baseline methods proposed in this paper, CoVis would become a prevalent DV-related task and inspire more studies in NLP for Data Mining direction.

6.2 Natural Language to SQL
Natural language to SQL (NL2SQL or text-to-SQL) is another closely related area that motivates text-to-vis development. Its ultimate goal is to enable databases with NL-based interfaces (NLIs). Past literature in this field first handles the single-turn text-to-SQL translation with representative studies such as Seq2SQL [49], SQLNet [52], TypeSQL [55], Syntax SQL [45], IRNet [15], [6] and [19] are two representative surveys about these studies. Moreover, numerous open-source text-to-SQL benchmarks have been released to promote research in this field, such as Spider [58] and WikiSQL [49]. Similarly, researchers also extend the vanilla text-to-SQL task to a multi-turn scenario with a public dataset such as SparC [59] and CoSQL [57]. The multi-turn interaction between the user and the NLI for DB systems would greatly improve the accuracy and further reduce the translation errors when converting NLQs to SQL queries, because it provides more chances to correct the errors in a sequence of conversations.

The success of the dialogue system and CoSQL also validates the necessity of multi-turn exchanges between machines and users in database scenarios. Our work is the first to study multi-turn exchanges between machines and users in the DV area.

6.3 Dialogue System
Tremendous efforts have been made by the community in designing various dialogue systems. The existing dialogue systems could be roughly divided into open-domain and task-oriented ones. The former focuses on providing chitchat or general responses to the users with typical systems like Microsoft’s Xiaoice [64], Alibaba’s AliMe [23], Google’s Meena [4, 5], and Meta’s BlenderBot [1]. Regarding implementation, retrieval-based and generation-based approaches were proposed for this task. The retrieval-based approach usually chooses the most appropriate instances from a library as the responses, whereas the generation-based method employs learning-based models to generate responses [21]. Recently, studies on combining both approaches were conducted to enjoy the merits of both methods [42, 53]. In contrast, the objective of task-oriented dialogue systems is typically to facilitate users to accomplish specific tasks in specific domains [10, 62]. Typical approaches to the task-oriented dialogue systems includes hybrid solutions [8] and the end-to-end ones [25, 51, 54]. In addition to textual dialogue systems, multi-modal dialogue systems are another direction that shows excellent usefulness in scenarios such as product recommendation in e-commerce [17, 26, 34, 38]. However, these multi-modal dialogue systems (e.g., MAGIC [34]) usually treat the product recommendation task as an image selection one from a candidate pool. Our task requires the model to generate the responses from scratch without any image candidate pool, which is much harder than selecting appropriate images from a large set of candidates. Thus, these existing multi-modal dialogue systems cannot perform well on this new proposed CoVis task. Our proposed CoVis task marries the popular DV task with dialogue system, and the proposed MMCVisNet model enlarges the family of multi-modal dialogue systems and would greatly alleviate the barriers to using DV systems for beginners and non-technical background users.

7 CONCLUSION
We introduce a new task named CoVis, aiming to generate data visualizations from conversations, and it could be used to construct interactive DV dialogue systems. To promote this field, we first construct a benchmark dataset, which includes a sequence of dialogues with diverse queries and responses. Then, we propose a multi-modal neural network named MMCVisNet to answer these DV-related queries. Experimental results have validated that MMCVisNet achieves superior performance over other existing baselines. This study is a good start in this new CoVis task. After we validate the feasibility of this task, there are many potential studies following this line of research. For example, a potential direction would be designing advanced neural structures like graph neural networks for modeling the connections in the dialogue context.

Acknowledgments We thank reviewers for their valuable comments and effort to improve this manuscript. Yuanfeng Song and Raymond Chi-Wing Wong is supported by fund WEB24EG01-H.


APPENDIX

A EVALUATION METRICS

We have a detailed definition about the evaluation metrics. The first set is to evaluate the quality of the textual-form response.

- **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:

\[
\text{BLEU} = \text{BP} \cdot \exp \left( \sum_{n=1}^{N} w_n \log \frac{p_n}{n} \right), \quad w_n = 1/n
\]

where \( p_n \) is the modified n-gram precision. The brevity penalty \( \text{BP} \) is calculated as:

\[
\text{BP} = \begin{cases} 
1 & \text{if } c > r \\
\exp \left( \frac{1}{1-c} \right) & \text{if } c \leq r 
\end{cases}
\]

where \( c \) is the length of 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 such as 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:

\[
\text{METEOR} = F_{\text{mean}}(1 - p),
\]

where \( F_{\text{mean}} \) is the harmonic mean of the unigram’s precision \( P \) and recall \( R \),

\[
F_{\text{mean}} = \frac{PR}{\alpha P + (1 - \alpha)R},
\]

where \( \alpha \) is a parameter between 0 and 1 with default value 0.9. The text segment’s penalty \( p \) function is defined as

\[
p = \gamma \left( \frac{c}{m} \right)^\beta, \quad \text{where } 0 \leq \gamma \leq 1,
\]

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

The second set is to evaluate the quality of the SQL-form response.

- **Sketch Accuracy**: This metric measures the percentage of matches between the structure of the predicted AST with the ground-truth one.

- **Overall Accuracy**: This metric reflects the percentage of matches between the final generated SQL query and the ground truth. To mitigate the impact of condition order, we further decompose each SQL statement into distinct components like "select column", "aggregator", and "condition". Rather than directly comparing the SQL string, we treat the condition part as a set and compare each element within the set.