

ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data

Can Liu¹ *Yun Han¹ †Ruike Jiang¹ ‡Xiaoru Yuan^{1,2,3} §

1) Key Laboratory of Machine Perception (Ministry of Education), and School of EECS, Peking University, Beijing, China

2) National Engineering Laboratory for Big Data Analysis and Application, Peking University, Beijing, China

3) Beijing Engineering Technology Research Center of Virtual Simulation and Visualization, Peking University, Beijing, China

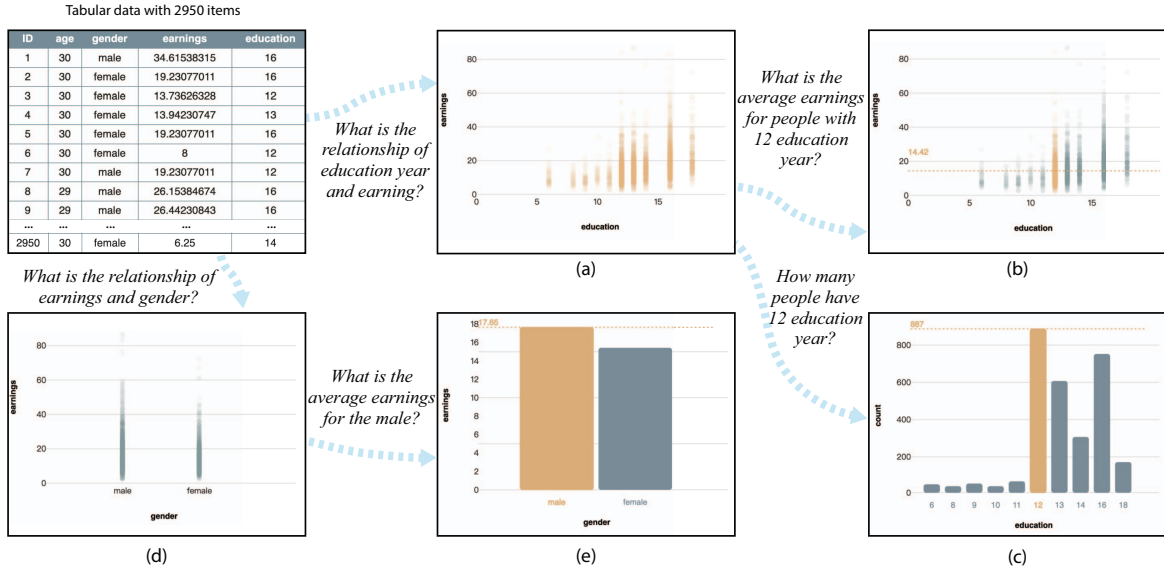


Figure 1: The tabular data and automatically generated visualizations by our method. The left-top shows the tabular data with 2950 items and 5 attributes, including earnings, education years, gender, etc. (a) to (e) are the generated visualizations for corresponding natural language questions.

ABSTRACT

We propose an automatic pipeline to generate visualization with annotations to answer natural-language questions raised by the public on tabular data. With a pre-trained language representation model, the input natural language questions and table headers are first encoded into vectors. According to these vectors, a multi-task end-to-end deep neural network extracts related data areas and corresponding aggregation type. We present the result with carefully designed visualization and annotations for different attribute types and tasks. We conducted a comparison experiment with state-of-the-art works and the best commercial tools. The results show that our method outperforms those works with higher accuracy and more effective visualization.

Keywords: Question answering, natural language, visualization, annotation, tabular data, machine learning, deep learning

* e-mail: can.liu@pku.edu.cn

† e-mail: yunhan@pku.edu.cn

‡ e-mail: jiangrk@pku.edu.cn

§ e-mail: xiaoru.yuan@pku.edu.cn (corresponding author)

1 INTRODUCTION

There is an increasing demand for users and companies to conduct data analytics. During the process of data analysis, instead of showing the data tables directly, generating data visualizations is a common approach to show the data features (e.g., distribution, outlier), because it is a more effective way to convey the features. Some tools (e.g., PowerBI) generates visualization according to the user-selected columns and rows and chosen visualizations. However, constructing corresponding visualizations requires expertise in both data and visualizations. On the one hand, users might not have enough pre-knowledge on which part of the data (data items and attributes) can fit their requirements. On the other hand, building proper visualization and annotation is non-trivial for the public without visualization design experience. Users need to consider which type of visualization can present the data well.

To lower the barrier for the public to construct visualization from data, natural language is a proper choice to present the requirements of the users. Therefore, several works take natural language as an interface to construct a visualization [4, 8, 17, 34], which largely reduced the barriers for users. However, the semantic parsing step of natural language in those works is based on simple word matching, which can not handle the problem of ambiguity and underspecification in the natural language. To achieve better accuracy, these works restrict the natural language to the commands or task in pre-defined templates to draw a visualization. Such pre-defined commands and tasks still require users to have the visualization design expertise to

decide how to operate and visualize the data. Take simple tabular data with two attributes, *year* and *oil production*, as an example to illustrate commands, tasks, and questions. “*Show a line chart of oil production and year*” is a command which clearly describes the attributes and visualization type. “*Show me the relationship of oil production and year*” is a task that describes the abstract visualization task. However, when observing a data table, the users may have some unrestricted questions about the data, which do not directly present as commands or task. For instance, “*Which year has the highest oil production?*”, “*Which year has the oil production between 200 to 300 mWh?*”, “*What is the trend of oil production across the year?*”, and so on. Obviously, questions have a larger coverage compared with tasks and commands and are more straightforward to present users’ direct thoughts comparing with tasks and commands. We defined a question without template and restrictions but can clearly present the requirement as an **open domain question**. If the open domain questions could be parsed precisely, visualization can be accessed by a large range of people because of the lower barrier in the open domain questions.

The previous works mentioned above can not handle the natural language queries when the queries do not fit into their templates or use the words match to the pre-defined list. There is a gap between the diversity in natural language and the visualization construction process. It urges us to explore the possibility of a generalized method rather than a template-based method. Approaches in deep learning show the potential to properly handle the open-domain natural language, inspiring us to bridge the gap using deep neural networks. Therefore, we propose a deep learning-based pipeline, ADVISor, to generate visualization with annotations for answering natural language questions on tabular data. Given tabular data and a natural language question as the input, we parse the data attributes and the question into vectors using a pre-trained language representation model. Several classification modules decide the data area (data items and attributes) and aggregation type (including summarization, average, extremes) according to the question and attribute vectors. The visualization and annotation type is decided by the chosen attributes and type of the attributes and the aggregation type. The carefully designed visualization and annotation are generated by visualizing and annotating the extracted data items and attributes.

To make the model be more generalized, we train the model using a dataset with a large volume of real-world tables and open-domain questions. WikiSQL [35] has 80654 human-annotated natural language questions on 24241 data tables from Wikipedia, which is one of the most massive question answering dataset for tabular data. Thus, we employ the WikiSQL dataset to train the semantic parsing model, which is proved by previous work to be enough to train our model with high accuracy. We then deployed the trained model into a demo system to show the visualization results for tabular data and corresponding questions. Once upon the selection of the data table, users can directly talk to our system to raise their questions and then get visualization answers with annotation in a few seconds.

To evaluate the effectiveness, we compared ADVISor with one of the state-of-the-art natural language interfaces for visualization construction works, NL4DV [17]. The comparison result shows that our model is more potent in parsing more flexible questions and show better visualization and annotation results. The contributions of ADVISor are the following:

1. We introduced a novel approach to generate visualization and annotations for tabular data and natural language questions with a deep learning-powered scheme.
2. We designed the proper visualizations with annotations according to the selected data area and corresponding aggregation types.

In the remainder of this paper, we briefly review the related work in Section 2, and introduce the method on tabular data in Section 4.

We provide the results in Section 6, followed by the limitations and the future work in Section 7. We conclude our paper in Section 8.

2 RELATED WORK

This section summarized the related research topics, including tabular data visualization, natural language interface for visualization construction, deep learning in natural language for visualization, and table-based question answering.

2.1 Tabular Data Visualization

Various visualization construction tools are proposed to support the visual exploration of tabular data. Polaris [26] allows users to drag and drop attributes to shelves defining encoding methods, and then visualizations are created using templates with encodings specified by users. Lyra [22] and iVisDesigner [21] have a more flexible and expressive visualization construction process, which makes visualization accessible to the public. Methods above use modular functions to construct visualizations, e.g., graphical objects, guide objects, generator objects in iVisDesigner and handles, and connectors in Lyra. Besides, there are some business tools such as Tableau [1] and ManyEyes [29] providing the means to generate, share, and publish visualizations without having to write any code. Wu et al. [31] proposed the semi-automated MuckRaker system for connecting news readers to a database of relevant context using a visualization interface. Other works focus on the generation and searching for huge tables of structured data but do not concern visualization [2].

The methods and tools mentioned above require users to specify the relationship between data attributes and visual channels, which may hinder novices from smooth visual data exploration. Thus, some works also focus on recommending possible visualizations. Keshif [33] focuses on automatically generating summaries of attributes by rules. SeeDB [28] recommends appropriate data attributes using deviation-based metric. Voyager [30] recommends a list of auto-generated charts based on some measurements for users to select to ease their burden of manually selecting attributes and encodings. However, our system can generate the visualization according to the task users provide directly.

2.2 Natural Language Interface for Visualization Construction

Natural language interaction for visualization has become a new method of interacting with data and visual analysis. Orko [25], Eviza [23], FlowSense [34], DataTone [8] have proposed methods for automatically generating visualizations in combination with natural language understanding. They automatically select data attributes through semantic analysis of user needs and generate corresponding visual charts or make changes to existing visual charts, such as filtering, highlighting, etc. However, the restrictions on user input here are relatively large, and they need to comply with some of the rules set by them.

Robust and complete natural language interface is difficult to achieve because they must deal with the problems inherent in automatically interpreting natural language. The reason is that natural language is often ambiguous and underspecified [24], requiring extensive parsing and complex reasoning. Nevertheless, natural language is becoming a promising interactive paradigm for visual data analysis, which can effectively improve visual systems’ usability. However, most of the current methods for automatically constructing visualizations based on natural language interaction have certain restrictions on natural language input. Deep learning-based natural language processing technology shows the potential to handle a more extensive range of natural language.

2.3 Deep Learning in Natural Language for Visualization

Several works have been devoted to generating corresponding natural language descriptions for visualizations. To address the difficulty of visually impaired people to read visualizations in bitmap form, Choi et al. [3] identify and describe the elements based on deep learning approaches. The approach focuses on the type and mapping relationship of the visualization and the direct presentation of the data. To improve the user's understanding of data features in visualizations, Liu et al. [15] proposed a method to automatically extract features from visualization charts and generate descriptions in natural language based on a deep learning model.

The natural language and the visualization often jointly together represent the content of the data. However, the user needs to switch between visualization and natural language descriptions and find the correspondence between them. To solve this problem, Lai et al. [14] proposed a method for automatic highlighting and annotating on a visualization based on textual descriptions. They use a deep neural network model combined with image processing techniques to extract and identify individual entity markers and their visual attributes in visual diagrams. Kim et al. [13] start from visual elements and natural language questions to get answers and generate the explainable processes. A deep learning-based algorithm is used to answer the transformed questions while interpreting the whole answer process based on templates. According to the experiments, the method can effectively generate effective explanations and answers in visual diagrams and natural language questions.

2.4 Question Answering for Tabular Data

Question answering (QA) is a research area that combines research from various fields, including Information Retrieval (IR), Information Extraction (IE), and Natural Language Processing (NLP). Among the various format that supports QA, tabular data is the most common data format. Traditional methods searched across tables and learned to perform aggregation operations according to the questions.

Khalid et al. [11] introduced a classifier to identify appropriate attributes and items by matching words of the natural language question. However, the matching method can only handle naive questions where the answers exist in the tables. To answer single relation questions, Cucerzan et al. [5] adopted a retrieval-based method [12] to answer the question for tabular data from the websites. Still, the methods mentioned above can not handle aggregation queries like count, sum, etc., which are common used by the public. Researchers attempted to parse the semantic of the natural language to logical forms to handle the complex natural language tasks precisely. Passport et al. [20] ranked the logical form candidates with a log-linear model and executed the highest-scoring one. Haug et al. [9] introduce a convolution neural network to select the logical form by joint embedding with the questions.

To solve the problems in open domain QA tasks without any predefined rules and templates, researchers proposed a growing number of end-to-end neural networks [16, 27]. Such end-to-end methods require large corpus of training data, which limits the scenarios. Neural programmer [18, 19] combines the semantic parsing method and end-to-end neural network, which inherits the benefits of two aspects. This method can get the state-of-art on table-based question answering without pre-defined templates and can handle complicated questions.

Our method combined the semantic parsing-based approach and the end-to-end neural network. To the best of our knowledge, our method is the first work that can handle a natural language without templates to construct visualizations and annotations.

3 PROBLEM STATEMENT

This paper aims to provide visualizations and annotations answer for open natural language questions on tabular data. The open questions

are raised by human users when observing the data, which are not restricted by pre-defined templates. The challenges come from parsing the natural language question to match the tabular data and generating correct visualization and annotation.

In the natural language part, matching the open domain question to the corresponding data content is a non-trivial task for the following reasons.

- The words in the question do not exactly match the attribute name in the table. For example, the attribute name is "population" whereas the word referring to that in the question is "number of people". In some extreme cases, the word in the question is not even a synonym for the attribute name. For instance, "since 2006" indicates the attribute "year"; the word "female" indicates the attribute "gender".
- Some questions contain filter operation on certain attributes. For example, users are looking for cars with more than 200 horsepower. In such a case, a filter operation, "> 200", is required in the horsepower attributes.
- Some questions indicate the aggregation operations. The user may ask how many people use the iPhone, which demands the aggregation operation, "COUNT", of items that meet the appropriate criteria. In this scenario, the aggregation (e.g., count, average, etc.) of the data is needed before generating a chart.

The above problems have long plagued researchers for constructing visualizations using appropriate natural language interfaces. Methods based on words searching and templates matching can only be effective to a very restricted fraction of natural language input from a human.

4 METHOD

In order to support open natural language questions, we propose a solution, ADVISor, which covers the three problems mentioned above correspondingly. The whole pipeline of ADVISor can be divided into the following steps, which are shown in Fig. 2.

1. **Natural Language Representation.** The natural language in the question and headers are unrestricted. Pretrained word representations provide a uniformed presentation vector of words, which can help to parse the semantic meaning of the words in question and headers. The synonyms would have similar representations, so the "number of people" will be closed to "population." The words that often occur together have close relationships in the vector, so the "since 2006" can indicate the "year".
2. **Related Data Area Extraction.** Finding the data area relevant to the question is a crucial part. For the tabular data, selecting the relevant attributes (columns) and filter conditions (rows) can determine the data area. In this part, we build deep modules for attributes selection and condition selection.
3. **Aggregation Type Identification.** Finding the corresponding aggregation tasks in the question is important for determining proper visualization type or annotation type. For example, the proper visualization for the question "how many Nobel Price winners are British?" shows the count of each country with a highlight on the British. And for the question "what is the average oil production of Asian countries", the "average" value will be shown using an annotated line on the generated chart.
4. **Visualization and Annotation Generation.** The visualization and annotation are generated according to the chosen attribute types and aggregation type. The annotation part is added to the visualization to highlight the answer by annotated line and highlight effect.

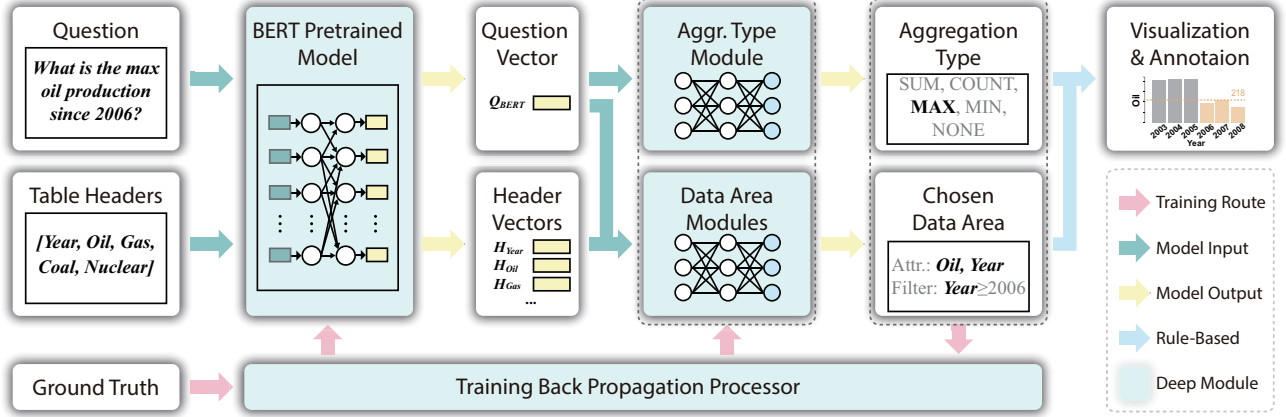


Figure 2: The pipeline of ADVISor. The first step is to convert the natural language in table headers and questions to vectors presenting the semantic meaning using the BERT model. Converted vectors are fed as input to decide the data area (including attributes and filter conditions) and the aggregation type. The visualization and annotation are generated according to the selected data area and the aggregation type.

In order to present the method precisely, we formally defined the content of each step. A tabular data \mathbf{T} is composed of M items (row) with N attributes (column). The header of a tabular data has N attributes names $H_j, j \in 1 \dots N$. Each attribute name H_j has T_j words: $H_j = \{h_k\}_{k=1}^{T_j}$. A natural language problem $Q = \{q_i\}_{i=1}^L$ consists of a series of words $\{q_i\}$, L is the number of words in the question.

4.1 Natural Language Representation

The question and headers are natural language with unrestricted words, i.e., any words may occur in the question and table headers. Pre-trained language representation model [6] has been shown to be effective for dealing with the unrestricted natural language, which provide a uniformed presentation vector of word. We fine tune the state-of-the-art work, BERT [7], which proves to be able to fine-tuned easily to create models for a wide range of tasks. BERT, short for bidirectional encoder representations from transformers, can produce a presentation vector and for the whole sequence.

In order to get a uniform presentation of the input, we combined the input question and headers to build a natural language sequence. With several special word [SEP] to separate the natural language question and the headers, similar to the setting of Hwang et al.’s work [10], which is presented as

$$[\text{CLS}], q_1, \dots, q_L, [\text{SEP}], h_{N,1}, \dots, h_{N,T_1}, [\text{SEP}], \dots, h_{N,1}, \dots, h_{N,T_N}, [\text{SEP}]. \quad (1)$$

where $h_{j,k}$ represents the k -th word of the j -th header, and T_j represents the total number of the j -th table header. We use the corresponding output vector of [CLS], H_{CLS} preserved the content of the whole question, and use the output of a header’s first word H_j to presents the j -th header.

4.2 Related Data Area Extraction

Finding the data part relevant to the question in tabular data is a crucial step for our work. For the tabular data, choosing the relevant attributes (columns) and filter conditions (rows) can decide the data area. For example, there is a table with attributes “year”, “country”, “population”. The question “What is the trend of the population in the USA since 2006?” relates to attributes population, country, and year. Among the three attributes, the population is the most crucial one since it is the result that the user directly wanted while the other two are on the filter conditions. The selected items have two filters including “country = USA” and “year > 2006”.

Several works [10,32] in the table question answering area extract the attributes and conditions and achieve very high accuracy. We regard the data area extraction following the state-of-the-art work [10] as a classification problem for the following tasks:

1. **Main attribute selection module:** Which attribute is the main attribute? The range of the classification is from 1 to N (the column size). The inputs are the encoded vector H_{CLS} and encoded header vectors $H_j, j \in 1 \dots N$. The output is to define which attribute is the main attributes. This is a typical classification task choosing 1 class from N classes. The probability of choosing column i as the main attributes is defined by following equations.

$$C_i = \sum_n \text{softmax}(H_i^T \mathbb{W} H_{CLS}) H_{CLS}$$

$$p_i = \text{Softmax}(\mathbb{W} \tanh([\mathbb{W} H_i : \mathbb{W} C_i]))$$

Here we take column-attention mechanism for each column. C_i is the context vector for question H_{CLS} given the i -th Headers H_i . The probability of i -th column, p_i , is calculated by the question H_{CLS} and the context vector of each header. Here, $[\cdot]$ denotes concatenation of two vectors; \mathbb{W} in different places indicate different affine transformations.

2. **Filters decision modules:** Using classifications to decide the number of the filters and choose the corresponding filter attribute. For each attribute, there are classification modules for determining what the value of the classification and the operation type (including $>$, $<$, and $=$) is. There are four parts in the filter decision modules, which decides the number of the filters, where attribute is the filter, the value, and the operation type. These modules are same with the “where-number”, “where-column”, “where-value”, “where-operator” module in previous works [10,32].

The inputs are the encoded vector H_{CLS} and encoded head vectors $H_j, j \in 1 \dots N$. And the outputs are formatted attributes and the filters. For example, the main attribute of the question “What is the trend of the population in the USA since 2006?” is “population” and the filters are “country = USA” and “year > 2006”.

4.3 Aggregation Type Identification

The aggregation type module’s goal is to determine the type of aggregation operation performed on the main attributes. For example,

the question “what is the average earnings of people with 12 years of education” have the aggregation type of average.

The common aggregation operations are “MAX”, “MIN”, “COUNT”, “SUM”, and “AVG”. Our task is to choose the best aggregations in the questions. Naturally, this task is taken as a classification problem by giving the questions input formation as input. Since the aggregation type is basically decided by the question, the input part of this model is H_{CLS} . The classification task is presented by

$$y_{agg} = \text{Softmax}(H_{CLS}\mathbf{W} + \mathbf{b}), \quad (2)$$

where $\mathbf{W} \in R^{d \times 6}$, $\mathbf{b} \in R^6$. Now the aggregation type is decided by the question information. For example, the aggregation type of “How many girls are in the school of EECS” for a table listing all students in a university is “COUNT”.

The loss function for the classification uses cross entropy, as shown below:

$$L_{Softmax} = - \sum_{i=1}^N Y_i \log y_i \quad (3)$$

where Y_i represents the i_{th} label after One-Hot encoding. y_i indicates the probability of i_{th} class predicted by the softmax layer of the model. In particular, the conditional column module is to perform binary classification on each column, so the objective function used is the binary classification cross-entropy loss function, as shown below:

$$L_{Sigmoid} = -\frac{1}{2}(Y \log y + (1-Y) \log(1-y)) \quad (4)$$

where Y indicates whether the attribute in the tabular data is an attribute in filter conditions. y indicates the prediction probability of the model after passing through the sigmoid layer. The total loss function of the model is the sum of the loss function of these modules.

4.4 Visualization and Annotation Generation

This section describes how to choose proper visualizations according to the extracted data area and the aggregation type. The questions of the tabular data are classified into three types.

1. The first type is the simple data retrieval question, which gets the certain cell’s value from the tabular data. For example, the question “Greece held its last Summer Olympics in which year?” refers to a certain answer. There are not many visualization needs for this kind of question.
2. The second type is the reasoning questions, whose results are calculated using several operations, including filter on certain conditions and aggregation on some results. For this kind of questions, visualization provides explanations for the answer, which makes the answer convincing.
3. The third type is the exploration question, whose result is ambiguous and uncertain. For example, “what is the relation of earnings and educations” belongs to this type. Visualization provides a basic view showing the results, which allows users to explore and find insights with their own knowledge.

Table 1 shows the corresponding examples for different types of questions for tabular data. We describe how we aggregate the data, choose the proper visualization type, and decide the annotation type for a detailed explanation.

Data Aggregation. We aggregate the original data according to data aggregation operation detected from the natural language input. For operation average and extreme (MAX, MIN), we divide the data into different groups according to the filter condition attribute.

We calculate the corresponding aggregation for each group. For example, to answer the question “what is the average publications of female students?”, the model will calculate the average value on the extracted quantitative attribute “publication” for each value in the categorical attribute *gender*. The result is that we generate the average value for males and females, respectively. For the operation count and sum, our method calculates the number or the sum of each choice and shows the results using a histogram.

Visualization Type Chosen. Given the data aggregation operation and attribute types, we introduce how we define rules for mapping them into visualizations.

The detailed rules for selecting visualizations can be seen in Table 2, where the number of every cell means the quantity of the corresponding attributes.

When no aggregation operation has been parsed from the natural language input, our approach directly presents the original data. For example, when there is only one categorical attribute, bar chart is used where each bar represents a data item. For ordinal attribute (with the same interval) and quantitative attribute, our approach uses line chart and scatter plot, respectively. When aggregation exists, we use aggregated charts for categorical or ordinal attributes, and unit charts along with annotation when there are only quantitative attributes. The illustration of visualization types according to different question types and attribute types is shown in Fig. 3.

Annotation Type Chosen. To make the answer convincing, our approach shows all the data item rather than showing only the filtered result. We annotate the visualizations to show both the filtering result and the answer to the question. To show the filtered data, we highlight the related elements in the visualization chart. For unit visualization, each highlighted element corresponds to a data item selected by the filtering condition. While for aggregation visualization, each highlighted element represents a data subgroup. To show the answer to the question, we draw an auxiliary line to show the final answer along with a text element showing the value.

5 IMPLEMENTATION

The system’s interface is shown in Fig. 4, which is separated into two views, a table view and a visualization view. The table view occupies the left part of the screen space, containing the table-related question, while the visualization view occupies the other half with the visualization answer to the question. Users can upload the table to the system. Corresponding questions can be input using typing or voicing. When the user clicks the generate button, the system can generate visualizations and annotations.

We designed the corresponding template for every visualization type of every task, which is built with HTML/CSS/JS technologies and the D3 framework for visualization.

6 RESULTS

In this section, we evaluate the accuracy of the trained model. We first describe the dataset used for training the model. To evaluate our system’s performance, we need to know whether our model can correctly extract the data area and identify the aggregation type and whether the visualizations match our expectations. Hence, we mainly focus on the following two aspects, the accuracy of problem types and sequence matching predicted by our model and the different visualization forms generated for different problem categories.

6.1 Dataset and Training Accuracy

To make the model generalized to a large range of natural language, a dataset covering various natural languages is needed. The corresponding data area and aggregation type is also needed in the dataset. The datasets in question answering for tabular data satisfy our requirements, which consists of (tables, questions, database query statements) triples. The main attribute, filter, and aggregation type can be found in the database query. An example of a database

Table 1: Question Type and Corresponding Examples

Overall Category	Question Type	Example
Exploration Question	Relation	What is the relationship of oil and gas?
	Question with distribution	What is the distribution of prestige across people with different publications?
Reasoning Question	Trend	How does the number of participating countries change since 2000?
	Extreme	What is the largest number of participating countries?
	Aggregation	How many events were in Athens, Greece?
	Sum	What is the total gold medals won by the top3?
Data Retrieval Question	Average	What is the average gold medals for countries which has more than 10 total medals?
	Simpleness	Greece held its last Summer Olympics in which year?

Table 2: Rules for Encoding Data Attribute (C for Categorical, O for Ordinal, and Q for Quantitative)

Aggregation	C	O	Q	Visualization	Annotation
None	1	0	1	Bar Chart	Highlight
	0	1	1	Line Chart	
	0	0	≥ 1	Scatter Plot	
Average	1	0	1	Bar Chart	Highlight + Line
	0	1	1	Line Chart	
	0	0	2	Scatter Plot	
Extreme	1	0	1	Bar Chart	Highlight+Line
	0	1	1	Line Chart	
	0	0	2	Scatter Plot	
Count	1-2	0	0	Bar Chart	Highlight
	0	1	0	Line Chart	
Sum	1-2	0	1	Bar Chart	Highlight
	0	1	1	Line Chart	

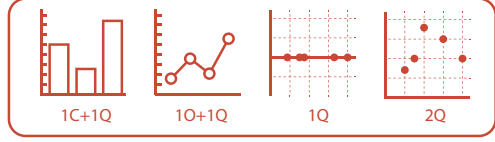
query is “SELECT AVG(oil) WHERE year > 2006”, which refers to the question, “What is the average oil production since 2006”. The main attribute is oil, the filter condition is year > 2006, and the aggregation type is AVG.

WikiSQL is one of the largest open public datasets of question answering for tabular data. WikiSQL¹ [35] contains 80,654 natural language questions on 24,241 tables crawled from Wikipedia with corresponding database query comments, which has recently attracted widespread attention. The purpose of collecting such a dataset is to help researchers build a deep neural network model to generate corresponding database query logical expressions (which contains data area and aggregation type) for a single table and a given natural language problem. Here is an example of WikiSQL. The question is “What is the lowest value for bronze with a total of 3 and a value for silver greater than 1 while the value of gold is smaller than 1?”. The headers are “Country”, “Gold”, “Bronze”, “Silver”, and “Total”. And the output is “SELECT MIN(Bronze) WHERE Total = 3 AND Silver > 1 AND Gold < 1”. The dataset is divided into the training set, validation set, and test set, containing 56355, 8421, and 15878 examples. The test set includes new questions raised for tabular data during the training process and new tabular data “unseen” during the training process.

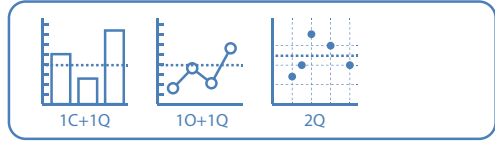
In our system, the problem’s type is directly consistent with the main attribute field in the database query statement, and the accuracy of the problem type classification is also directly related to the performance of the selection aggregation module. We can use the main attribute field to verify the accuracy $Accuracy_{cls}$ of the task type of users’ question. In addition, we also need to analyze the mapping accuracy from natural language questions to the database query to evaluate the entire system’s performance. WikiSQL has

¹WikiSQL: <https://github.com/salesforce/WikiSQL>

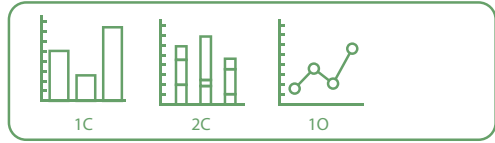
Relation, Distribution, Trend



Average, Extreme



Count



Sum

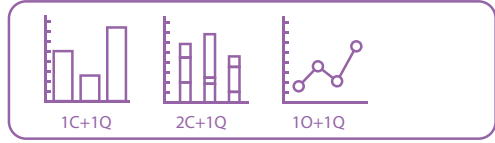



Figure 3: Visualizations and annotations for different attribute types and annotation types. (C for Categorical attribute, O for Ordinal attribute, Q for Quantitative attribute)

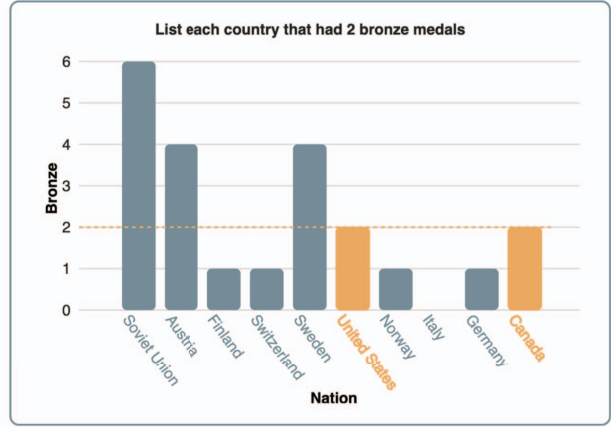
designed two evaluation indicators to measure the performance of the model, which are the accuracy of logical expression mapping $Accuracy_{lf}$ and the accuracy of executing logical expression results $Accuracy_{ex}$.

$Accuracy_{lf}$ represents the accuracy of the database query predicted by the model and the dataset. $Accuracy_{ex}$ refers to the accuracy of results executed by the database query predicted by the model and the results labeled in the dataset. One disadvantage of $Accuracy_{ex}$ is that we can construct a database query that does not correspond to the question but still obtain the same results. For example, if no two people with different names and their “SSN” is “123”, then the execution results of “Select COUNT (name) Where SSN = 123” and “Select COUNT (SSN) Where SSN = 123” will be the same. Moreover, the design of the template for generating visual charts is completely based on the fields of database query, so we mainly focus on the accuracy of logical expression mapping

Automatic Answer and Visualization Generation for Tabular Data

Index	Rank	Nation	Gold	Silver	Bronze	Total
1	1	Soviet Union	7	3	6	16
2	2	Austria	4	3	4	11
3	3	Finland	3	3	1	7
4	4	Switzerland	3	2	1	6
5	5	Sweden	2	4	4	10
6	6	United States	2	3	2	7
7	7	Norway	2	1	1	4
8	8	Italy	1	2	0	3
9	9	Germany	1	0	1	2
10	10	Canada	0	1	2	3

Question: List each country that had 2 bronze medals?  Generate



Answer: United States, Canada

Figure 4: Interface of the our demo system. The left part shows the tabular data and the input box, which allows users to input via typing or voice. The right part shows the visualizations answering the questions.

$Accuracy_{If}$. The accuracy is calculated as follows:

$$Accuracy_{cls} = \frac{N_{cls}}{N} \quad (5)$$

$$Accuracy_{If} = \frac{N_{If}}{N} \quad (6)$$

where N is the total number of samples in the dataset, N_{cls} is the number of samples with the correct classification of the sample analysis type, and N_{If} is the database query logic. The number of samples in the expression is exactly the same as in the dataset. Our model is evaluated on the test set after training on the training set, with a total of 15878 examples. We focus on the classification accuracy of the model in the type of user analysis tasks, so we separately measured the model’s accuracy in selecting the aggregation operation classification results. There are six types of selection aggregation operations in the data, namely “MAX”, “MIN”, “COUNT”, “SUM”, “AVG” and “NONE”. Our model gives 13544 correct predictions on the analysis task type, and the results show that the classification accuracy of the model in this test set is 85.3 %.

The accuracy of logical expression mapping is also an important evaluation index of this dataset. When evaluating the accuracy of the logical expression mapping, we did not consider the order of the conditional sequence, and the accuracy of the final model reached 83.1%. Excluding the mark errors that exist in some data items, we think this result is quite impressive.

6.2 Case 1: Energy Production Dataset

We use the tabular data about energy production in Table 3 as a case to evaluate the system. The tabular data is statistical data of energy production in a country, measured in megawatt-hours (MWh) per person per year. Here are three examples of different question types, respectively: “What is the highest nuclear production across the year?”, “What is the trend of oil production since 2004?”, and “What is the relation between gas and oil?”. Our system then generates a bar chart, line chart, and scatterplot according to these questions. The results are shown in Fig. 5. The bar chart (a) shows the nuclear production of these years and highlight the highest position with a line. Users may also be interested in the trend of oil

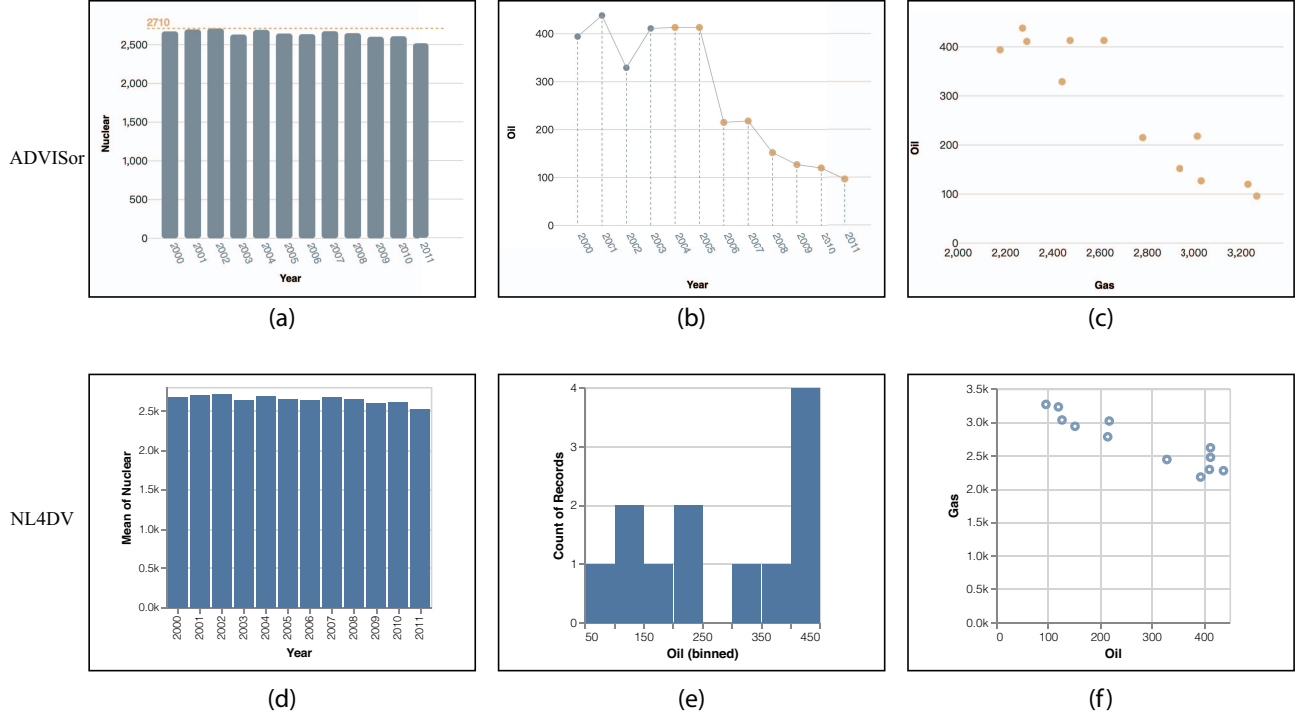
Table 3: Statistical data of energy production

Year	Population(M)	Coal	Oil	Gas	Nuclear
2000	282.17	6968	394	2179	2672
2001	285.08	6679	438	2274	2697
2002	287.80	6717	329	2441	2710
2003	290.33	6798	411	2292	2631
2004	293.05	6751	413	2475	2691
2005	295.75	6806	413	2618	2644
2006	298.59	6666	215	2782	2636
2007	301.58	6686	218	3018	2674
2008	304.38	6524	152	2939	2649
2009	307.01	5719	127	3034	2602
2010	309.33	5972	120	3230	2609
2011	313.85	5523	96	3267	2518

production. The system shows the line chart (b) of the oil production with the highlights on the selected years. When users are interested in two attributes’ relationship, such as scatterplot in (c) is produced. From the results above, our system shows the ability of handling various questions with proper visualizations and highlights.

We also conduct a comparison with the state-of-the-art work NL2DV [17]. NL2DV proposes a Python Library for generating a visualization given tabular data and corresponding natural language queries. The format of the input is similar to our work. From the (d), (e), and (f) of the Fig. 5, we found that except for the third question, our model outperforms the NL4DV because of more powerful semantic parsing techniques and annotations. There are mainly the following difference:

1. The filter conditions parsing is failed in the NL4DV. And there is no annotation in the visualization generated by NL4DV. So in the first question, there is no highlight in the highest bar.



What is the highest nuclear production across the year?

What is the trend of oil production since 2004?

What is the relation between gas and oil?

Figure 5: The generated visualizations and annotations for different question on the oil dataset. (a), (b), and (c) are the visualization generated by ADVISor, while (d), (e), and (f) are the visualization generated by NL4DV [17].

- The semantic parsing is not flexible enough in NL4DV. Some words that not directly match the attributes name is not correctly parsed. (b) and (e) shows ADVISor can deduce the year attribute from the phrase “since 2004” while NL4DV can not. Therefore, in the second question, NL4DV generates a wrong visualization.

6.3 Case 2: Data with Large Number of Items

ADVISor also works well with large tabular data and works well in real-world scenarios. It is difficult for the user to explore tabular data with thousands of items.

In this example scenario, Jack wants to analyze the tabular data comes from March Current Population Surveys (CPS) with data on the relationship of earnings and education ², which has 2950 items with five attributes, including ID, age, gender, income, and years of education. The table is displayed in the left-top part of the Fig. 1.

After observing the data headers, Jack is first interested in the relationship between earnings and education years. So he asks, “What is the relationship of earnings and education years?”. The system presents a scatter plot (a), showing the relationship between these two attributes. He finds that higher education indicates higher income, but there is an outlier in the education years of 12, which seems to be higher than 13. He thinks this may be due to the large number of people who have education years of 12. So he asked the system, “How many people have 12 years of education?” The system comes up with results (e), which confirms his hypothesis that 12 years is precisely the highest value. Further, he moved on to ask, “What is the average earning for people with 12 education years?”

²CPSSWEducation: vincentarelbundock.github.io/Rdatasets/datasets.html

The system also gave the correct answer (b), highlighting 12 years of education in the chart and giving the corresponding auxiliary line to show the average dendrites.

There was also a column in the data showing gender. Again, he wanted to see the impact of different genders on income, so he asked the system, “What is the relationship of earnings and gender?”. The system generates a scatter plot of salary and gender (c). He then asks what the average salary is for men, so he asks, “what is the average earnings for the male?”. The system gives a histogram (d) and highlights the average salary of male.

The results in this scenario show our model’s ability to handle the open questions raised by humans and generate the proper visualizations and annotations to present and highlight the answer to the questions.

6.4 User Exploration

We conducted an experiment that asks users to explore tabular data in Sec 6.3 using natural language questions. We recruited 8 participants (age 21-25; 2 female; 6 male; graduates). The participants are required to ask more than five questions after observing the data. We told them that the age attribute here is trivial since it only has two values: 29 and 30. We did not provide any templates for them, and we encourage them to ask any objective questions that can be answered using the data.

We collected 40 non-repeating questions based on the data. Participants are interested in the relationship between earnings and education (5 of 40). For example, “What is the relationship between earnings and education?”, “Do those people with higher education years have higher earnings”, and “What is the trend of earnings with different education years”. Participants are also interested in

the relationship of gender and earning, earnings of a certain gender, and the comparison of two genders. For example, “*Is the average income of men higher than that of women?*”, “*What is the income of males?*”.

Participants might not use the words that occurred in the table headers. For example, the word gender has not occurred in the question. The participants use men (man), male(s), women (woman), female(s) in their question. In some question, the attributes “earning” is presented as “income” (16 of 40), “make money”(3 of 40), and “earns”(10 of 40). For example, “*How much do males earn?*”, “*What is the average income of women?*”.

ADVISor generates reasonable results for 72.5% questions(29/40). The failure cases include unrecognized aggregation type and missing attributes. For example, for the question “What is the median income of women”, the median aggregation is not in our training dataset, ADVISor finds the “average” number rather than “median”. In some cases, the ADVISor fails to recognize all the attributes in the question. For example, ADVISor only recognizes the attribute “earnings” from the question “How do earnings change as the education year increases?”. The reason is that there is not much sentence that begins with “how” in the WikiSQL datasets.

NL4DV got only 17.5% reasonable results (7/40) based on the same questions. The failure cases mainly because of the implicit attributes in the question. For example, participants tend to use spoken language like “income” and “make money” to present their requirements. In some cases, NL4DV also recognized the words that do not exactly match with the headers. Word-stemming is used before the matching, so the word “earn” can also match to “earnings”. However, the word matching in NL4DV may introduce new errors. For example, the word “year” is matched to “age” in question like “do those with higher education years have higher earnings”.

ADVISor outperforms NL4DV in some cases because of the implicit attributes. Generally, ADVISor allows generating proper visualization for a larger range of natural language questions.

7 DISCUSSION

Our approach offers a promising example of how to automatically generate related visualizations with annotations for open-domain question on tabular data. The positive results in Section 6 confirm the benefits of ADVISor. This work fill the gap between the public without visualization experience and the meaningful visualization. Users can get meaningful insights from the visualization by simply ask questions about the tabular data. Still, there are some limitations of our works.

7.1 Limitation in Semantic Parsing

Currently, our model consists of two parts, the semantic parsing, and the visualization generation. There are some limitation of natural language process model due to the diversity and ambiguity natural language processing.

For example, questions involve multiple computational processes, which means the questions with arithmetics, such as “*Who had more silver medals, Cuba or Brazil?*”. There is more than one main attribute in this question. The current model can not handle such questions containing more than one attribute. Therefore, such questions are failed using the current model setting,

Some errors in the semantic parsing can be corrected by user interaction [8].

7.2 Limitations in Datasets

Another aspect that influences our ability is that the current dataset is designed for data to retrieve questions, whose questions have the exact answers, i.e., the answer refers to one or several values in the tabular data. There are only have limited aggregation type in the dataset, e.g., count, average. For example, the question “what is the average oil production of America since 2006” is inside the dataset,

while the question without an exact answer is not inside the dataset, e.g., “what is the trend of the oil production.” Though ADVISor can generate some visualization for the ambiguous question containing “trend” and “relations” thanks to the generalization ability of the semantic parsing module, the model may fail for more complex questions.

The reason is that such a dataset (e.g., WikiSQL) is built to solve the problem of converting natural language to the database query. The existing datasets do not contain such an exploration type of questions. For better supporting the visualization construction for questions, producing a dataset designed for visualization tasks is needed.

7.3 Limitations for Visualization

The design of the visualization generation steps proposed in this paper is mainly based on the experiences gained through the visualization practice. We defined the visualization and annotation rules for different attributes and aggregation types. However, it is not realistic for us to travel through all the possibilities.

Visualization construction is a complex task that considering more than the attributes and aggregation type. More factors may influence the effect of visualization. For example, when there are more than two filter conditions in the question, how to filter, visualize, and highlight the data is non-trivial. It is also possible to explore uniform frameworks for choosing visualization and annotation automatically in our future study.

In the future, we plan to explore deep learning approaches for choosing visualization and annotation automatically. For example, after collecting the path when users use our system, a model can be trained to learn how to construct the visualization for different input questions.

7.4 Future Work

Apart from generating a visualization once for a time, we can also consider the multi-run question answering for tabular data. In the real scenario, the users explore the generated visualizations and have some findings in the visualization. More questions can be raised according to the result of the previous step. The new question’s result can be directly shown using highlights in the existing visualization or transition to the new visualization. The animations and the consistent view will help the user have a consistent mental map when exploring the data.

8 CONCLUSION

This paper proposes an automated pipeline, ADVISor, for generating the visualization with annotations as the answer when given the tabular data and question. We use the pre-trained language presentation model for converting the natural language in question and the table headers to corresponding vectors. Deep neural networks are introduced to take the vectors of question and table headers as input and extract the data area and the aggregation type. The natural language question is answered with carefully designed visualization and annotations. With the visualization aids, this system can enhance understanding of the answer to the question and enable a more comprehensive understanding of the tabular data question.

We compare ADVISor with state-of-the-art work NL4DV [17]. The results show that our method can handle flexible questions with the more powerful natural language parsing. The user study also demonstrates that our method outperforms the commercial tool by providing better visualization for the question.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable comments. This work is supported by NSFC No. 61872013.

REFERENCES

- [1] C. W. Barnes. Conservative coupling between modes of propagation—a tabular summary. *Proceedings of the IEEE*, 52(1):64–73, Jan 1964.
- [2] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proceedings of VLDB Endowment*, 1(1):538–549, 2008.
- [3] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmqvist. Visualizing for the Non-Visual: Enabling the visually impaired to use visualization. *Computer Graphics Forum*, 38(3):249–260, 2019.
- [4] K. Cox, R. E. Grinter, S. L. Hibino, L. J. Jagadeesan, and D. Mantilla. A multi-modal natural language interface to an information visualization environment. *International Journal of Speech Technology*, 4:297–314, 2001.
- [5] S. Cucerzan and E. Agichtein. Factoid question answering over unstructured and structured web content. In *Proceedings of the 14th Text REtrieval Conference*, 2005.
- [6] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Proceedings of 29th Conference on Neural Information Processing Systems*, pages 3079–3087, 2015.
- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, June 2019.
- [8] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 489–500, 2015.
- [9] T. Haug, O. Ganea, and P. Grnarova. Neural multi-step reasoning for question answering on semi-structured tables. In *Proceedings of the 40th European Conference on IR Research*, pages 611–617, 2018.
- [10] W. Hwang, J. Yim, S. Park, and M. Seo. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*, 2019.
- [11] M. A. Khalid, V. Vijkoun, and M. de Rijke. Machine learning for question answering from tabular data. In *Proceedings of the 18th International Workshop on Database and Expert Systems Applications*, pages 392–396, 2007.
- [12] D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth. Question answering via integer programming over semi-structured knowledge. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1145–1152, 2016.
- [13] D. H. Kim, E. Hoque, and M. Agrawala. Answering questions about charts and generating visual explanations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020.
- [14] C. Lai, Z. Lin, R. Jiang, Y. Han, C. Liu, and X. Yuan. Automatic annotation synchronizing with textual description for visualization. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020.
- [15] C. Liu, L. Xie, Y. Han, D. Wei, and X. Yuan. Autocaption: An approach to generate natural language description from visualization automatically. In *Proceedings of the 2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 191–195, 2020.
- [16] Z. Lu, H. Li, and B. Kao. Neural enquirer: learning to query tables in natural language. *IEEE Data Engineering Bulletin*, 39(3):63–73, 2016.
- [17] A. Narechania, A. Srinivasan, and J. Stasko. NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, 2021.
- [18] A. Neelakantan, Q. V. Le, M. Abadi, A. McCallum, and D. Amodei. Learning a natural language interface with neural programmer. *arXiv preprint arXiv:1611.08945*, 2016.
- [19] A. Neelakantan, Q. V. Le, and I. Sutskever. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*, 2015.
- [20] P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 1470–1480, 2015.
- [21] D. Ren, T. Höllerer, and X. Yuan. iVisDesigner: Expressive interactive design of information visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2092–2101, 2014.
- [22] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. *Computer Graphics Forum*, 33(3):351–360, 2014.
- [23] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th Annual ACM Symposium on User Interface Software and Technology*, pages 365–377, 2016.
- [24] V. Setlur, M. Tory, and A. Djalali. Inferencing underspecified natural language utterances in visual analysis. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 40–51, 2019.
- [25] A. Srinivasan and J. Stasko. Orko: Facilitating multimodal interaction for visual exploration and analysis of networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):511–521, 2017.
- [26] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.
- [27] S. Vakulenko and V. Savenkov. TableQA: Question answering on tabular data. In *Posters and Demos Track of the 13th International Conference on Semantic Systems*, 2017.
- [28] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Automatically generating query visualizations. *Proceedings of VLDB Endowment*, 7(13):1581–1584, Aug. 2014.
- [29] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. M. McKeon. Manyeyes: a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007.
- [30] K. Wongsuphasawat, D. Moritz, A. Anand, J. D. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, 2016.
- [31] E. Wu, A. Marcus, and S. Madden. Data in context: Aiding news consumers while taming dataspace. In *Proceedings of the 1st VLDB Workshop on Databases and Crowdsourcing*, pages 47–50, 2013.
- [32] X. Xu, C. Liu, and D. Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017.
- [33] M. A. Yalçın, N. Elmqvist, and B. B. Bederson. Keshif: Rapid and expressive tabular data exploration for novices. *IEEE Transactions on Visualization and Computer Graphics*, 24(8):2339–2352, 2018.
- [34] B. Yu and C. T. Silva. Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–11, 2020.
- [35] V. Zhong, C. Xiong, and R. Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, abs/1709.00103, 2017.