NL2VIS Business Intelligence System

Liwei Ye
lye474@connect.hkust-gz.edu.cn
Hong Kong University of Science and Technology(GZ)
Guangzhou, Guangdong, China

ABSTRACT

The proliferation of digital data in the fourth industrial revolution has created unprecedented opportunities for businesses to derive insights from vast datasets. However, the complexity of data analytics often poses a barrier for professionals lacking technical expertise. To address this challenge, we present the development of a Natural Language to Visualization (NL2VIS) Business Intelligence (BI) system. This innovative system empowers users to generate data visualizations through intuitive natural language queries. Leveraging advancements in natural language processing (NLP) and visualization technologies, NL2VIS simplifies the interaction between users and structured data, making data analysis more accessible to business experts without a background in data science. The system integrates a user-friendly web interface utilizing Vue.js for dynamic front-end interactions and Flask for efficient back-end processes. We discuss the challenges encountered, such as data interpretation and the identification of appropriate visualization types, as well as the solutions implemented to overcome these hurdles. Our key contributions include the design of an interactive front-end layout and an intelligent back-end capable of processing and visualizing data based on user input. Looking ahead, we outline potential future enhancements, including domain-specific language processing, user intent prediction, interactive visualization features, and personalized user experiences through adaptive learning. The NL2VIS system stands to revolutionize the field of BI by making data-driven decision-making more intuitive and aligned with natural human communication.

CCS CONCEPTS

• **Information System**; • **Human-centered computing**;

KEYWORDS

Business Intelligence (BI), Natural Language Processing (NLP), Natural Language to Visualization (NL2VIS), User Interface (UI), User Experience (UX), Domain-Specific Language Processing

ACM Reference Format:
Liwei Ye. 2024. NL2VIS Business Intelligence System. In . ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 MOTIVATION

In the era of the fourth industrial revolution, digitalization permeates various industries, transforming information into data stored within digital devices, such as computers and smartphones. Cloud platform has become the predominant medium for data storage and sharing, revolutionizing how business manage data.

Data has emerged as a crucial asset for businesses, prompting a shift in focus towards uncovering the underlying wealth of information. This has led to the rise of data scientists, professionals skilled in deciphering patterns within data. Nonetheless, data scientists might encounter a hurdle when striving to comprehend and capitalize on data-driven insights due to the potential lack of domain-specific knowledge possessed by business experts, despite their analytical prowess.

To bridge this gap between business experts and data, we introduce an innovative solution that facilitates automatic data visualization through a Business Intelligence (BI) system. It involves aggregating and analyzing substantial data volumes from diverse sources, including internal systems, customer interactions, and market trends. The overarching objective of BI is to empower business users and provide actionable information that enables companies to fine-tune their operations, boost performance, and secure a competitive edge.

With these considerations in mind, we propose the development of an NL2VIS Business Intelligence system. This system will allow users to effortlessly translate natural language queries into precise visualizations on a web interface, streamlining interactions with structured data and offering a user-friendly solution for data analysis.

2 RELATED WORK

Business Intelligence (BI) systems have evolved to leverage sophisticated visualization tools and database technologies, enhancing the way users interact with data. Conventional tools such as D3[16], Vega-Lite[8], and VizQL underpin platforms like Tableau[7] and Microsoft Power BI[6], which offer user-friendly interfaces for data visualization. These systems have democratized data analysis to some extent by enabling users without deep technical skills to generate visual insights[14].

On the database front, technologies like Tableau's Hyper DB and the integration of D3 with Apache Spark via the Falcon project have addressed the challenge of visualizing large-scale datasets. This has allowed for real-time data processing and visualization, even with vast amounts of data.

While tools like Excel[4], Google Sheets[2], and Oracle Data Visualization Desktop[5] offer built-in visualization functions, and platforms such as IBM DB2 [3]and Amazon QuickSight[1] have expanded these capabilities, there are still significant limitations. Users often face a steep learning curve, and the tools may lack

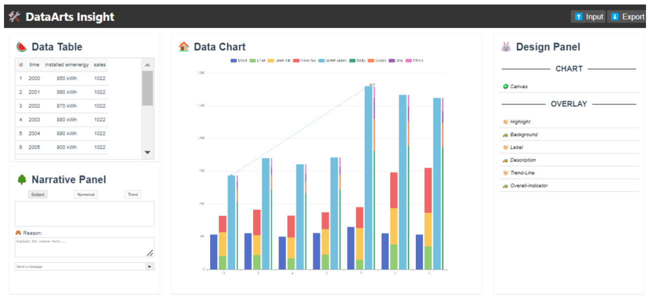


Figure 1: System Overview

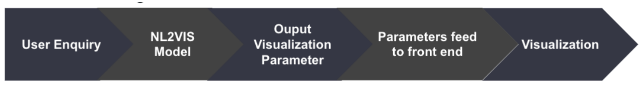


Figure 2: NL2VIS Pipeline

comprehensive natural language processing (NLP) capabilities for intuitive data exploration.

The performance issues associated with real-time visualization of very large datasets also persist, which can impact the overall user experience. These limitations highlight the need for more accessible, natural language-driven BI systems that can simplify data analysis for all users, a gap that the NL2VIS Business Intelligence System aims to fill.

3 DESIGN OF METHOD AND SYSTEM

3.1 Framework

Our NL2VIS BI system offers a more user-friendly alternative to conventional BI systems, especially for non-technical users. Users can input requests in natural language on the web page, and the system generates data visualizations accordingly.

The NL2VIS BI platform leverages Vue.js[11] for the front-end, creating an interactive and modular experience. The back-end utilizes Flask[9], a Python web framework, ensuring smooth communication between the front-end and server processes.

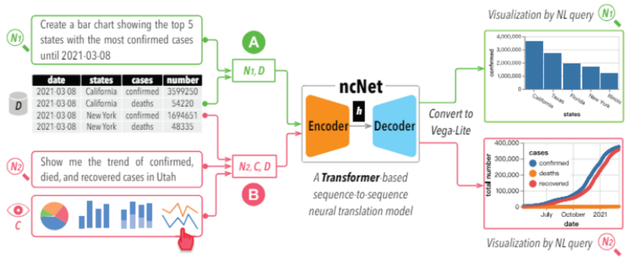
Vue.js is a progressive JavaScript framework optimized for building user interfaces. It focuses on the view layer, offering an adaptable and efficient approach to constructing single-page applications. Flask provides the tools and libraries necessary for building scalable web applications, supporting various extensions and middleware.

The front-end comprises the main App.vue component, which imports additional Vue components such as the Header Panel, Data Table, Data Chart, Design Panel, and Input Box.

In the back-end, a Large Language Model processes user input, returning data and visualization parameters to the front-end for graph display.

3.2 NL2VIS

The core of effective data visualization lies in the transformation of raw data into visually informative representations using the most suitable types of charts. This critical process often requires human insight and decision-making to ensure the visualizations accurately


N1
Create a bar chart showing the top 5 states with the most confirmed cases until 2021-03-08
D
N2
Show me the trend of confirmed, died, and recovered cases in Utah
C
A
N1, D
B
N2, C, D
ncNet
Encoder
h
Decoder
A Transformer-based sequence-to-sequence neural translation model
Convert to Vega-Lite
Visualization by NL query N1
confirmed
states
California
Texas
Florida
New York
Illinois
Visualization by NL query N2
total number
cases
confirmed
deaths
recovered
July
October
2021
date

date	states	cases	number
2021-03-08	California	confirmed	3599250
2021-03-08	California	deaths	54220
2021-03-08	New York	confirmed	1694651
2021-03-08	New York	deaths	48335

Figure 3: NCnet Architecture

convey the underlying data insights. This is precisely where the NL2VIS model plays a crucial role in bridging the gap between natural language queries and meaningful visualizations.

The "Natural Language to Visualization by Neural Machine Translation" approach introduces a transformative concept in data visualization through the ncNet model, a sophisticated Transformer-based architecture designed specifically for translating natural language queries into visual representations of data. This innovative model opens up new possibilities for interpreting complex data queries and generating corresponding visual outputs with enhanced accuracy and relevance.

The NL2VIS framework operates on a synthesis methodology that involves converting NL2SQL benchmarks into NL2VIS benchmarks. This conversion process entails translating SQL query structures into precise visualization specifications, while also making necessary adjustments to natural language descriptions.

By harmonizing SQL query structures with visualization requirements and refining natural language descriptions, the NL2VIS framework creates a comprehensive benchmark dataset. This dataset serves as a valuable resource for advancing research in the realm of natural language-driven data visualization, paving the way for innovative developments in this field(see Figure 2 for the NCnet Architecture)[13].

3.3 Nl2SQL+NL2Chart (Natural language to chart form)

In our system design, we have adopted a holistic approach to enhance the capabilities of our natural language to visualization (NL2VIS) system by integrating Nl2SQL and NL2Chart functionalities. This comprehensive strategy involves refining the existing NL2VIS model for improved performance, fine-tuning the NL2SQL model to enhance its accuracy, and expanding the system's capabilities with the introduction of SQL2VIS functionalities. This addition allows for an alternative route to data visualization post data inquiry, further enriching the system's offerings.

The integration of NL2VIS, NL2SQL, and SQL2VIS components is illustrated in the accompanying flowchart (see Figure 4). By synergizing these components, our system is well-positioned to offer enhanced interpretability and usability in converting user queries into visually insightful representations. This integrated approach not only streamlines the data visualization process but also empowers users to extract meaningful insights from their queries through intuitive and interactive visualizations.

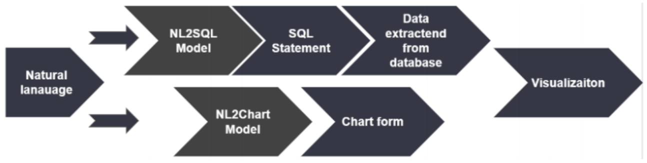
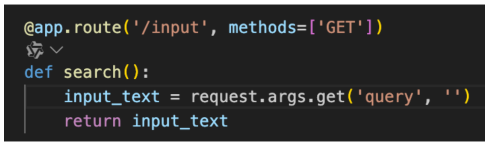
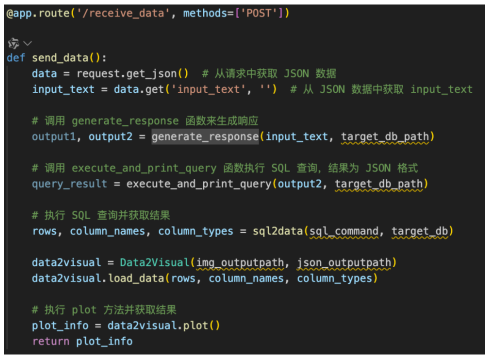


Figure 4: NL2VIS + NL2Chart Workflow



```
@app.route('/input', methods=['GET'])
def search():
    input_text = request.args.get('query', '')
    return input_text
```

Figure 5: Input Box Back-End



```
@app.route('/receive_data', methods=['POST'])

def send_data():
    data = request.get_json()  # 从请求中获取 JSON 数据
    input_text = data.get('input_text', '')  # 从 JSON 数据中获取 input_text

    # 调用 generate_response 函数来生成响应
    output1, output2 = generate_response(input_text, target_db_path)

    # 调用 execute_and_print_query 函数执行 SQL 查询，结果为 JSON 格式
    query_result = execute_and_print_query(output2, target_db_path)

    # 执行 SQL 查询并获取结果
    rows, column_names, column_types = sql2data(sql_command, target_db)

    data2visual = Data2Visual(img_outputpath, json_outputpath)
    data2visual.load_data(rows, column_names, column_types)

    # 执行 plot 方法并获取结果
    plot_info = data2visual.plot()
    return plot_info
```

Figure 6: Data Visualization Back-End

4 KEY CONTRIBUTIONS

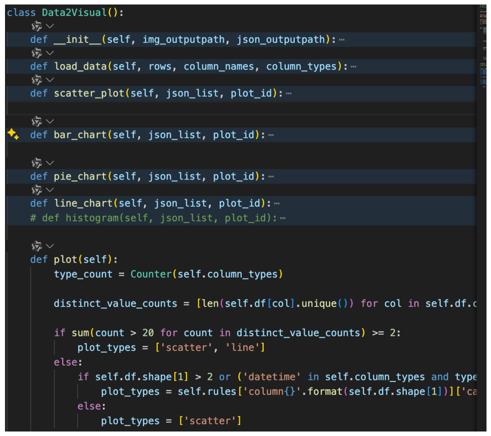
In this project, my primary responsibilities encompass front-end and back-end development of the auto visualization, alongside dedicating efforts to refining an NL2Chart model.

4.1 Back-End Development

4.1.1 Query Processing. Our back-end introduces an optimized input handling mechanism where user queries, submitted via a GET request's 'query' parameter, are seamlessly integrated into the data retrieval pipeline.

This design ensures that user inputs are efficiently channeled to the back-end for further processing.

4.1.2 Data Visualization Engine (send-data function). Central to our solution is the send-data function, which orchestrates the transformation of SQL query outputs into rich visual insights. Leveraging the sql2visual.py module, this function interprets the SQL results generated by the NL2SQL model, applies intricate logic outlined in the Data2Visual class, and assembles comprehensive visualization blueprints. These blueprints, encapsulating chart specifics such as type, axes labels, chart titles, and raw visualization data, are compiled into a JSON object named "plot info". Subsequently, they are dispatched to the front-end via a POST request, facilitating the seamless delivery of dynamic visualizations.



```
class Data2Visual():
    def __init__(self, img_outputpath, json_outputpath): ⋯
    def load_data(self, rows, column_names, column_types): ⋯
    def scatter_plot(self, json_list, plot_id): ⋯

    def bar_chart(self, json_list, plot_id): ⋯

    def pie_chart(self, json_list, plot_id): ⋯
    def line_chart(self, json_list, plot_id): ⋯
    # def histogram(self, json_list, plot_id): ⋯

    def plot(self):
        type_count = Counter(self.column_types)

        distinct_value_counts = [len(self.df[col].unique()) for col in self.df.c

        if sum(count > 20 for count in distinct_value_counts) >= 2:
            plot_types = ['scatter', 'line']
        else:
            if self.df.shape[1] > 2 or ('datetime' in self.column_types and type
                plot_types = self.rules['column{}'.format(self.df.shape[1])]['ca
            else:
                plot_types = ['scatter']
```

Figure 7: Data2Visual Class

4.1.3 Data2Visual - SQL2VIS. The cornerstone of our back-end innovation is the Data2Visual class, meticulously crafted to dynamically generate visualizations tailored to the structure and nature of the underlying dataset.

The Data2Visual script defines:

Data Processing. Data is cleaned and formatted into suitable structures, including natural language queries, SQL queries, and corresponding chart specifications. Data augmentation techniques are applied to enhance dataset diversity and model robustness.

Visualization Generation. The script defines classes and functions for generating different types of visualizations, such as scatter plots, bar charts, pie charts, line charts, and histograms. Rules and logic are implemented to determine the appropriate visualization type based on the data characteristics.

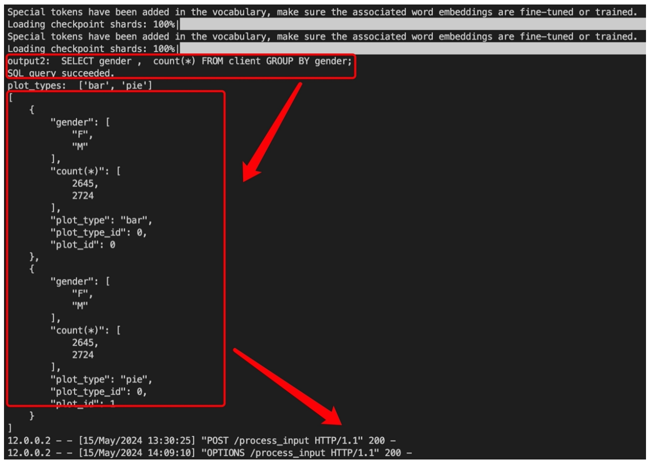
This intelligent system categorizes datasets and selects the most appropriate visualization technique based on a sophisticated set of rules embedded in its methods.

For datasets with two categorical columns, **scatter plots** are generated. Numeric indices are identified and used to plot data points on the graph. The script iterates through different combinations of numeric columns to create scatter plots, capturing relationships between variables.

Bar charts are generated when there is a mix of numeric and categorical data. The script identifies numeric and categorical indices to create bar charts that visualize the distribution of data across categories.

Pie charts are generated based on the dataset structure. Single-column datasets result in pie charts illustrating the distribution of unique values. Two-column datasets lead to pie charts showing relationships between two variables.

Line charts are generated for datasets with numeric data suitable for trend analysis. The script identifies numeric and categorical indices to plot data points over a continuous scale, highlighting trends over time or categories.



```
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
Loading checkpoint shards: 100%|
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
Loading checkpoint shards: 100%|
output2:  SELECT gender ,  count(*) FROM client GROUP BY gender;
SQL query succeeded.
plot_types:  ['bar', 'pie']
[
    {
        "gender": [
            "F",
            "M"
        ],
        "count(*)": [
            2645,
            2724
        ],
        "plot_type": "bar",
        "plot_type_id": 0,
        "plot_id": 0
    },
    {
        "gender": [
            "F",
            "M"
        ],
        "count(*)": [
            2645,
            2724
        ],
        "plot_type": "pie",
        "plot_type_id": 0,
        "plot_id": 1
    }
]
12.0.0.2 - - [15/May/2024 13:30:25] "POST /process_input HTTP/1.1" 200 -
12.0.0.2 - - [15/May/2024 14:09:10] "OPTIONS /process_input HTTP/1.1" 200 -
```

Figure 8: NL to SQL to Plot Input

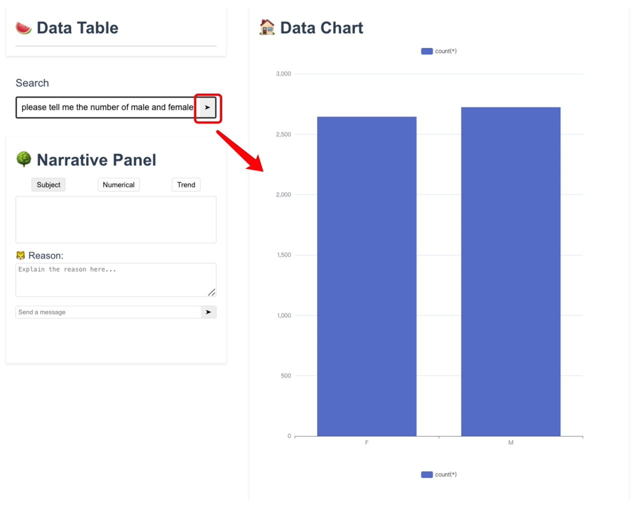
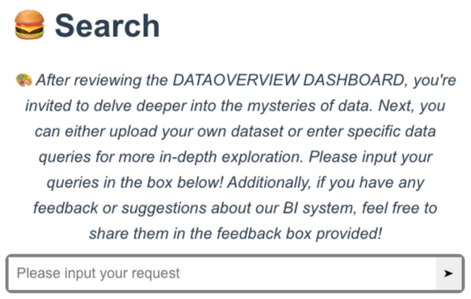


Figure 9: NL2VIS Procedures

These rules are conditional on the availability and types of columns within the dataset, and the class rules defined at initialization (‘self.rules‘) guide which chart types are applicable for datasets with different compositions of categorical and numeric columns. The ‘plot‘ method orchestrates these by determining the applicable chart types based on column counts and then invoking respective plotting functions.

By dynamically applying these rules informed by column types and dataset dimensions, Data2Visual ensures the generation of insightful visual narratives that align with the data’s inherent characteristics.

An illustrative workflow begins with the user providing natural language input. Upon clicking the "Send" button, the visualization process is triggered, and the NL2SQL model initiates the conversion of natural language to an SQL query. Subsequently, our Data2Visual script generates the plot input parameters, as illustrated in Figure 8.



🍔 **Search**

🎨 *After reviewing the DATAOVERVIEW DASHBOARD, you're invited to delve deeper into the mysteries of data. Next, you can either upload your own dataset or enter specific data queries for more in-depth exploration. Please input your queries in the box below! Additionally, if you have any feedback or suggestions about our BI system, feel free to share them in the feedback box provided!*

Figure 10: Input Box

For instance, a sample NL input such as "Please tell me the number of male and female clients" will prompt the corresponding visualization to appear under the Data Chart section. The model then utilizes the NL2SQL model to translate the natural language into SQL. An SQL query like "SELECT COUNT(client ID), gender FROM client GROUP BY gender," depicted in Figure 8, yields a structured output showcased in Figure 9, where a bar chart is generated accordingly. Figures 11 and 12 are produced using different SQL queries.

4.2 Front-End Development

My primary focus encompasses part of front-end and full back-end development. The front-end features the Data Chart and Input Box as core components. The Data Chart displays visualizations which is converted from the natural language input.

4.2.1 User Input Box. The input box functionality within the system serves as a pivotal user interface element, facilitating the submission of natural language queries or instructions. When users interact with the input box by entering their queries and triggering the action, the system initiates a series of backend processes. The input NL language query is swiftly transmitted to the backend for processing, leading to the automatic generation of visualizations tailored to the interpreted query.

Leveraging sophisticated natural language processing techniques, the back-end extracts essential information from the query to dynamically produce insightful visual representations of the data. These visualizations are then presented to the user, enabling interactive exploration and the extraction of valuable insights based on the initial natural language input.

4.2.2 SQL2VIS Integration- DataChart. The DataChart component embodies the visual manifestation of the SQL2VIS concept. Following the conversion of natural language queries into SQL and subsequent data retrieval, E-charts technology is employed to render engaging visualizations.

Our system dynamically supports a quartet of chart types – bar, pie, scatter, and line charts – each selected dynamically by the backend based on the retrieved data’s characteristics. This ensures that users are presented with the most informative and intuitive visual representation of their data, enhancing decision-making capabilities and fostering a deeper understanding of complex datasets.

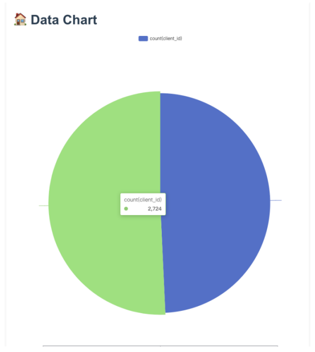


Figure 11: Auto Generated Pie Chart

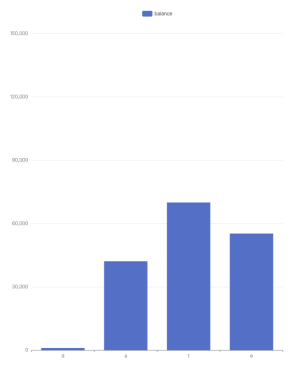


Figure 12: Auto Generated Bar Chart

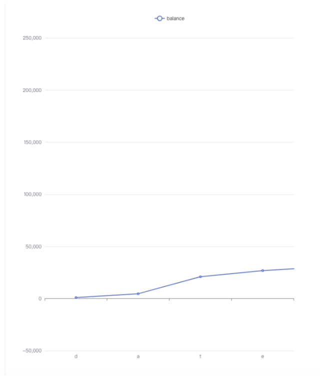


Figure 13: Auto Generated Line Chart

4.3 LLM fine-tuning

4.3.1 Data Preprocessing Pipeline. In our data preprocessing pipeline aimed at optimizing the nvBench dataset for refinement, we meticulously prepared the data to ensure seamless integration with the LLaMA2-7b model. This involved a multi-step process.

Data Cleansing. We rigorously eliminated errors and discrepancies within the dataset to maintain its integrity and reliability.

Table 1: Results of fine-tuning of NL2Chart

Model	Dataset	Before FT	After FT
LLAMA2-7b	nvBench	62.3	98.5
T5	nvBench	57.9	89.7

Schema Alignment. The dataset was meticulously aligned to incorporate natural language queries, SQL queries, and corresponding chart specifications in a structured format tailored to meet the model's input requirements.

Data Augmentation. To enhance dataset diversity and robustness, we augmented the dataset with synthetic examples covering outlier scenarios and less common query/chart combinations.

4.3.2 Low-Rank Adaptation (LoRA). In the fine-tuning phase, we employed the Low-Rank Adaptation (LoRA) methodology to enhance the LLaMA2-7b model using the refined nvBench dataset for the NL2Chart task. By integrating trainable low-rank matrices into the transformer layers, LoRA optimized the model's self-attention and feed-forward networks. This optimization significantly boosted the model's performance in generating precise and contextually relevant chart visualizations from SQL query outputs.

The strategic parameter updates facilitated by LoRA ensured the retention of the model's foundational knowledge, enabling it to effectively adapt to the distinct requirements of new tasks. Additionally, our automated dynamic visualization generation process streamlined chart type selection based on SQL query results and prompts. This automated approach ensured accurate data representation through automated field mapping and real-time visualization command execution, enhancing the system's efficiency in platforms like Vega-Zero.

5 CONCLUSION

In conclusion, the development of the NL2VIS Business Intelligence system represents a significant step forward in democratizing data analytics and visualization. By bridging the gap between complex data and business expertise, this system unlocks the potential for more informed decision-making across various industries. Our approach leverages the simplicity of natural language to interact with and visualize data, reducing the barriers traditionally faced by non-technical professionals.

The integration of a user-friendly web interface, powered by Vue.js, and a robust back-end, supported by Flask, ensures a seamless and interactive experience. This system not only facilitates the generation of dynamic visualizations in response to user queries but also promotes an intuitive understanding of data trends and insights.

Throughout the design and development process, we encountered and overcame numerous challenges, particularly regarding data interpretation and visualization selection. These experiences have informed our approach and will guide future enhancements to the system. Our contributions to both the front-end and back-end of the NL2VIS system lay the foundation for a more accessible and efficient tool for data analysis.

6 FUTURE WORK AND LIMITATION

One limitation of our NL2VIS BI system pertains to the partial automation of visualization generation. At times, the system may fail to produce a meaningful or valid visualization autonomously, necessitating human intervention for data interpretation. This dependency on human judgment arises from the complexity of certain datasets that may require preprocessing steps like binning or other transformations to render them suitable for accurate visualization. As a result, the system's ability to generate visualizations seamlessly is impeded by the need for manual data preprocessing and interpretation, highlighting a key limitation in achieving fully automated visualization processes.

In future endeavors, our focus will be on enhancing the model by refining it based on the ncnet model structure and expanding its capabilities for code generation, particularly for platforms like echarts. Additionally, we aim to augment the system with a more interactive approach, potentially integrating a conversational system. This enhancement will enable a deeper understanding of the desired graph outputs and goals, fostering improved communication and alignment between user expectations and system performance.

6.1 Natural Language Understanding Enhancements

6.1.1 Domain-Specific Language Processing. To ensure that our NL2VIS Business Intelligence system effectively serves users across various industries, we plan to enhance the natural language processing (NLP) component to better interpret domain-specific jargon. This enhancement will involve training our NLP models on specialized corpora that include industry terminology, enabling the system to recognize and process the unique language used in different business contexts[15]. This improvement will also address the challenge of ambiguities and complex queries that may arise due to the specialized nature of business language.

6.1.2 Contextual Embedding and Transfer Learning. We propose to leverage advanced NLP techniques, such as contextual embedding, to capture the meaning of words in the context of surrounding text. By utilizing models such as BERT or GPT-3, which provide deep contextual representations, our system will gain a more nuanced understanding of user queries[10]. Furthermore, we aim to apply transfer learning methods to adapt pre-trained models to our specific application, enhancing their ability to comprehend natural language queries related to data visualization.

6.2 User Intent Prediction

6.2.1 Predictive Modeling for User Intent. To improve the responsiveness of the NL2VIS system, we plan to implement machine learning models capable of predicting a user's intent. These models will analyze patterns in previous interactions, query logs, and user behavior to anticipate the user's needs. By accurately predicting intent, the system can proactively generate relevant visualizations, thereby enhancing the user experience.

6.3 Interactive Visualizations

6.3.1 Enhanced Data Chart Interactivity. The Data Chart component will be enhanced to support a higher degree of interactivity, allowing users to engage with visualizations by drilling down into specifics, applying filters, and manipulating data directly within the visualization[12]. This capability will empower users to explore their data more thoroughly and intuitively, leading to a deeper understanding of the underlying trends and patterns.

6.3.2 Dynamic Data Updates and Animations. Incorporating real-time data updates and animations into the NL2VIS system will create a more dynamic and engaging user experience. As the underlying data changes, the visualizations will update automatically, reflecting the most current information. Animations will guide the user's attention to changes or updates within the visualization, providing a seamless transition between data states.

6.4 Personalization and User Profiles

6.4.1 Creation of Personalized User Profiles. We envision a system where personalized user profiles are central to the user experience. These profiles will store individual preferences, frequently used datasets, and custom visualizations, streamlining the workflow for return users. Personalized profiles will allow users to quickly access their preferred settings and visualization types, saving time and enhancing productivity.

6.4.2 Adaptive Learning from User Interactions. The NL2VIS system will be designed to learn from individual user interactions to provide a tailored experience that improves over time. Machine learning algorithms will analyze usage patterns to adapt the system's behavior and recommendations to each user's unique requirements. This adaptive learning approach will lead to a more personalized and efficient user experience, as the system becomes more attuned to each user's data visualization needs.

REFERENCES

[1] Amazon quicksight: Cloud-based business intelligence. https://aws.amazon.com/quicksight/. Accessed 31 Dec 2018.
[2] Google sheets: Free online spreadsheets for personal use. https://www.google.com/sheets/about/. Accessed 31 Dec 2018.
[3] Ibm db2. https://www.ibm.com/analytics/db2. Accessed 31 Dec 2018.
[4] Microsoft excel. https://products.office.com/en-us/excel. Accessed 31 Dec 2018.
[5] Oracle data visualization desktop. https://docs.oracle.com/en/middleware/bi/data-visualization-desktop/tutorials.html. Accessed 31 Dec 2018.
[6] Power bi: Interactive data visualization bi tools. https://powerbi.microsoft.com. Accessed 31 Dec 2018.
[7] Tableau. https://www.tableau.com. Accessed 31 Dec 2018.
[8] Vega: A visualization grammar. https://vega.github.io/vega/. Accessed 31 Dec 2018.
[9] Armin Ronacher. Flask: A python microframework. *Pocoo Team*, 2010. Accessed on 2021-09-28.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.
[11] Evan You. Vue.js. https://vuejs.org/, 2014. Accessed on 2021-09-28.
[12] Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky. A tour through the visualization zoo. *Communications of the ACM*, 53(6):59–67, June 2010.
[13] Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*, page 13, Virtual Event, China, June 2021. ACM.
[14] Xuedi Qin, Yuyu Luo, Nan Tang, and Guoliang Li. Making data visualization more efficient and effective: a survey. *The VLDB Journal*, 28(5):707–746, 2019.
[15] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. OpenAI Blog, 2018.
[16] Hadley Wickham. ggplot2 – elegant graphics for data analysis. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2009.

Received 10 May 2024; revised 15 May 2024; accepted 20 May 2024