International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2024, 14(1): 57–72, doi: 10.21655/ijsi.1673-7288.00321
©2024 by Institute of Software, Chinese Academy of Sciences. All rights reserved.

International Journal of Software and Informatics

Research Article

Navi: Data Analysis System Powered by Natural Language Interaction

Yupeng Xie (谢宇鹏)[1], Yuyu Luo (骆昱宇)[2], Jianhua Feng (冯建华)[3]

[1] (Department of Computer Technology and Applications, Qinghai University, Xining 810016, China)
[2] (The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511400, China)
[3] (Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)
Corresponding author: Jianhua Feng, fengjh@tsinghua.edu.cn

Abstract With the advent of the big data era, the significance of data analysis has increasingly come to the forefront, showcasing its ability to uncover valuable insights from vast datasets, thereby enhancing the decision-making process for users. Nonetheless, the data analysis workflow faces three dominant challenges: high coupling in the analysis workflow, a plethora of interactive interfaces, and a time-intensive exploratory analysis process. To address these challenges, we introduce with this paper Navi, a data analysis system powered by natural language interaction. Navi embraces a modular design philosophy that abstracts three core functional modules from mainstream data analysis workflows: data querying, visualization generation, and visualization exploration. This approach effectively reduces the coupling of the system. Meanwhile, Navi leverages natural language as a unified interactive interface to seamlessly integrate various functional modules through a task scheduler, ensuring their effective collaboration. Moreover, in order to address the challenges of exponential search space and ambiguous user intent in visualization exploration, we propose an automated approach for visualization exploration based on Monte Carlo tree search. In addition, a pruning algorithm and a composite reward function, both incorporating visualization domain knowledge, are devised to enhance the search efficiency and result quality. Finally, we validate the effectiveness of Navi through both quantitative experiments and user studies.

Keywords data analysis; data query; visualization; natural language; Monte Carlo tree search

Citation Xie YP, Luo YY, Feng JH. Navi: Data analysis system powered by natural language interaction, *International Journal of Software and Informatics*, 2024, 14(1): 57–72. http://www.ijsi.org/1673-7288/321.htm

In the era of big data, data analysis showcases the significance in a variety of fields such as business intelligence and scientific research[1]. However, the current data analytics workflow still faces challenges such as high coupling in the analytics workflow, a plethora of interactive interfaces, and a time-intensive exploratory analytics process.

As shown in Figure 1, the mainstream data analytics workflow includes the following key steps: (1) users query a subset of data from the database for analytics through SQL queries or

This is the English version of the Chinese article "Navi: 基于自然语言交互的数据分析系统. 软件学报, 2024, 35(3): 1194–1206. doi: 10.13328/j.cnki.jos.007074"
Funding items: National Natural Science Foundation of China (61925205, 62232009, 62102215); Science and Technology Research and Development Program of China Railway Corporation (K2022S005)
Received 2023-07-17; Revised 2023-09-05; Accepted 2023-10-24; IJSI published online 2024-03-29

Python; (2) users probe into the data insights in the dataset through data tables or visualizations and use the visualization results for presentation and communication; (3) during the visualization process, users use visualization query languages (e.g. Vega-Lite[2]) or interactive visualization tools (e.g. Tableau[3]) to generate visualization results. If the visualization results cannot meet the user's need for data analytics, the user may repeat the above steps.

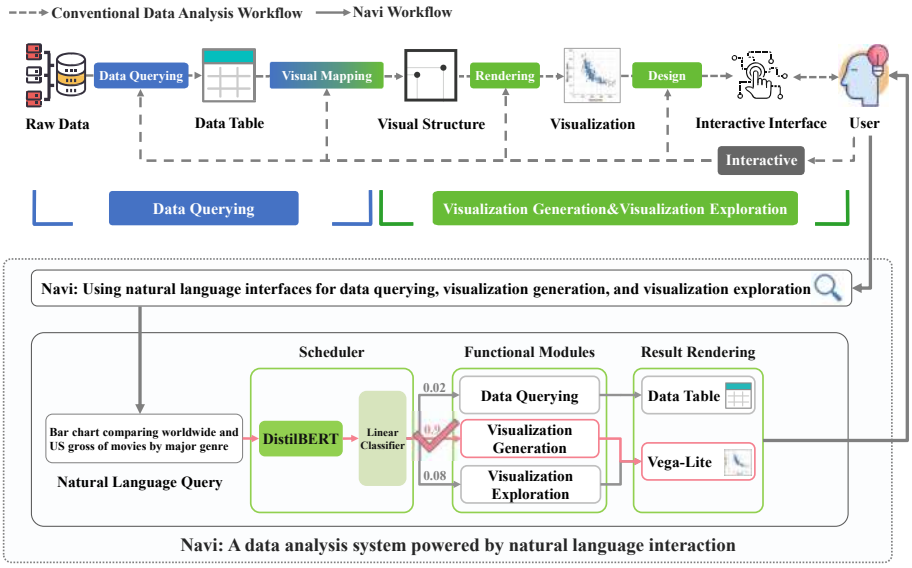


Figure 1 Framework of Navi as a data analysis system powered by natural language interaction

The steps above in the data analytics workflow are interdependent and usually require repeated iterations for exploratory analytics, which result in high coupling in the analytics workflow and high time consumption. In addition, data analytics, as a professional and complex task, poses high requirements for professional skills and visualization analytics abilities of users.

With the aim of reducing the threshold and time required for users to conduct data analytics and enhancing the quality and efficiency of data analytics, researchers have optimized the data analytics workflow to improve the efficiency of human-machine collaboration. As shown in Table 1, the available systems mainly optimize single links in the data analytics workflow, failing to simultaneously address the three major challenges of high coupling in the analytics workflow, a plethora of interactive interfaces, and a time-intensive exploratory analytics process.

Table 1 Comparison between Navi and existing systems

Systems	Data querying	Visualization generation	Visualization exploration
RATSQL[4]	√	×	×
SEQ2VIS[5]	×	√	×
DeepEye[6]	×	×	√
KG4VIS[7]	×	×	√
Navi	√	√	√

To address the above challenges, we propose the following objectives (Figure 2).

(1) Modular analytics workflow. To improve the flexibility and reusability of the data analytics workflow, we abstract three modules from the mainstream data analytics workflows to reduce the coupling in the analytics workflow. The modular design enables the flexible utilization of different data querying, visualization, and interaction methods to meet the diverse and customized data analytics needs.

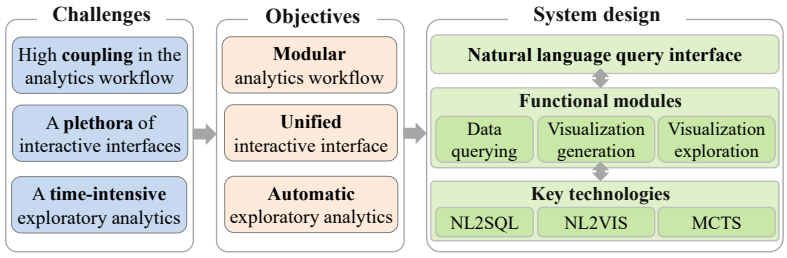


Figure 2 Objectives and system design of this paper

(2) A unified interactive interface. Data analytics is a complex process involving multiple types of tasks, each with a specific interactive interface, and users need to convert between different tasks. To address the challenge of diverse types of interactive interfaces, we leverage natural language as a unified interactive interface to integrate diverse data analytics tasks (such as data querying, visualization generation, visualization exploration), thus improving the collaboration efficiency.

(3) Automatic exploratory analytics. Exploratory analytics is a key step in data analytics. Conventional methods require users to generate and select visualizations and optimize the analytics results through iterations. The above process is cumbersome, complex, time-consuming, and requires high professional skills of users. To address these problems, we propose an automated approach for visualization exploration based on Monte Carlo Tree Search (MCTS)[8], which combines data features, visualization domain knowledge, and user preferences to automatically explore datasets and generate valuable visualizations without excessive user intervention and feedback. The automatic approach can improve the efficiency and quality of exploratory analytics.

To achieve the above objectives, we design and implement Navi, a data analysis system powered by natural language interaction (Figure 1). Navi adopts a modular design philosophy that abstracts three core functional modules: data querying, visualization generation, and visualization exploration, from mainstream data analytics workflows[9] (achieving objective (1)). With natural language as a unified interactive interface, a task scheduler is designed to parse natural language queries input by users and distribute them to corresponding functional modules (achieving objective (2)). As shown in Figure 2, each module of Navi is implemented based on deep learning. The key technology of the data querying module is to convert natural language to SQL (NL2SQL), while that of the visualization generation module is to convert natural language to visualization (NL2VIS). For the visualization exploration module, an approach based on MCTS is proposed to efficiently explore the visualization space (achieving objective (3)). Therefore, Navi serves as a data analysis system powered by natural language interaction, achieving data querying, visualization generation, and visualization exploration. The main contributions of this paper are summarized as follows.

(1) The modular analytics philosophy is adopted to abstract three common modules from the mainstream data analytics workflows, reducing the coupling in the analytics workflow (Section 1).

(2) Natural language is leveraged as a unified interactive interface to integrate data querying, visualization generation, and visualization exploration modules (Section 2).

(3) An automatic approach for visualization exploration is proposed based on MCTS to achieve automatic exploration and visualization of datasets. A pruning algorithm and a composite reward function, both incorporating visualization domain knowledge, are devised to enhance the search efficiency and result quality (Section 3).

(4) With the approaches above, this study designs Navi, a natural language-powered end-to-end data analysis system and validates its effectiveness through experiments (Section 4).

1 Introduction of Navi

1.1 Modular design

Adopting a modular design philosophy, Navi is composed of three functional modules: data querying, visualization generation, and visualization exploration (Figure 3). The modular design reduces the coupling in the analytics workflow. The function of each module is described as follows.

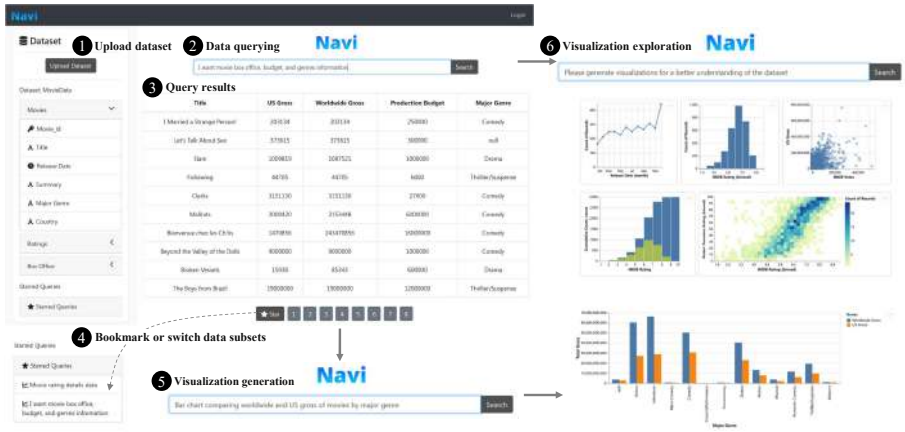


Figure 3 Overview of Navi

- Data querying module. This module can convert natural language into SQL and return query results (Figure 3③). By selecting and connecting data tables, as well as filtering, sorting, and aggregating data, this module can return the data subsets of interests to users, providing a convenient data query experience. Furthermore, users can bookmark or switch data subsets for further exploration based on the data subsets to be analyzed (Figure 3④).
- Visualization generation module. This module analyzes user intents based on the input natural language queries and generates visualization results based on data features and analytics objectives (Figure 3⑤).
- Visualization exploration module. This module mainly assists users in understanding datasets. It can integrate data features to generate multiple visualization results and display the data insights (Figure 3⑥). This module is applicable for scenarios with ambiguous user intent and provides users with a starting point for analytics.

This modular design endows Navi with flexibility, expandability, and compatibility. Regarding the flexibility, users can choose different functional modules according to their needs and achieve seamless switch between these modules, which can complete the entire data analytics task through collaboration. In terms of the expandability, Navi can replace or update a module at any time to adapt to technological advancements. The compatibility of Navi is embodied in the functional interoperability with other systems embracing the same interactive interface by reusing modules.

1.2 Task scheduler

The task scheduler can integrate various functional modules. It can parse the natural language queries input by users and distribute them to the corresponding functional modules. This section takes the task scheduler as a multi-classifier and introduces its implementation steps.

- Dataset building. We build a dataset containing a large number of natural language queries and corresponding module labels for training and validating the task scheduler. This dataset covers diverse types of queries, including NL2SQL, NL2VIS, and visualization exploration. We extract and label natural language queries from the NL2SQL benchmark Spider[10] and the NL2VIS benchmark nvBench[5]. At the same time, the natural language queries in nvBench are converted into the queries applicable for the visualization exploration module, which are more concise and vague and do not contain clear visualization types, sorting requirements, or other detailed information.
- Model training. The task scheduler uses a pre-trained DistilBERT[11] model as the basic model, which is based on the architecture of Transformer[12] and has fewer parameters and lower computational complexity but excellent performance. We use AutoTokenizer from Hugging Face Transformers to convert the query into the DistilBERT input format and divides the dataset into training, validation, and testing sets. To build a task-specific scheduler, we add a linear classifier with three output units on the basis of DistilBERT, employs cross entropy loss as the loss function, and uses the AdamW[13] optimizer with weight decay for weight updating.

With the task scheduler, Navi becomes comprehensive and user-friendly. Navi flexibly integrates functional modules to meet the needs of users for data querying, visualization generation, and visualization exploration. Moreover, it uses natural language queries as the interactive interface, simplifying user operations and achieving a dialogue-style exploration with data.

2 Natural Language-powered Data Querying and Visualization Generation

Data querying and visualization generation are essentially the same type of tasks which convert input sequences (such as natural language queries) into output sequences (such as SQL or Vega Lite[2] queries). We adopt a Sequence to Sequence (Seq2Seq) model[14] to complete such tasks.

The Seq2Seq model generally consists of an encoder and a decoder and leverages different neural networks. As shown in Figure 4, the task of the encoder is to comprehend the input sequence and generate a small vector h to represent the input, while that of the decoder is to generate a series of outputs based on h. In this work, natural language queries as input are a sequence $[q_1, q_2, \cdots, q_l] \in V_{\text{in}}$ composed of tokens (or words), and SQL or Vega-Lite[2] queries as output are also a sequence $[y_1, y_2, \cdots, y_k] \in V_{\text{out}}$ composed of tokens, where V_{in} and V_{out} represent the vocabularies of input and output, respectively. Further, we introduce the Transformer-based encoder and decoder[12].

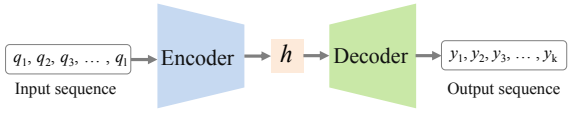


Figure 4 Seq2Seq model

- Transformer-based encoder. We connect the input sequence $n_V = [q_1, q_2, \cdots, q_l]$ with the database table schema information $A = [a_1, a_2, \cdots, a_m]$. The input sequence after

connection is

$$X = [x_1, x_2, \cdots, x_n] = [q_1, q_2, \cdots, q_l, a_1, a_2, \cdots, a_m] \tag{1}$$

where $n = l + m$. We then use a pre-trained global word embedding (GloVe)[15] to map each token x_i to its vector representation. After token embedding is complemented, the embedded token is input into the bi-directional Transformer network which outputs a series of encoding vectors:

$$h = [h_1, h_2, \cdots, h_n] \tag{2}$$

- Transformer-based decoder. The decoder also adopts a Transformer-based architecture and introduces the attention mechanism[16]. According to the hidden state h, the decoder generates an output sequence:

$$Y = [y_1, y_2, \cdots, y_k] \tag{3}$$

At each time step t, the decoder predicts the SQL query token y_t based on the current state S_i, previous token, and attention vector C_i.

We only need to adjust the parameters of this model and then use a large number of input/output sequence pairs related to the task as the training data to train the data querying and visualization generation modules, respectively. Specifically, the data querying module is trained based on the natural language/SQL query (NL, SQL) pairs in the NL2SQL benchmark Spider[10]. For example, for a natural language query input of "fetch all COVID-19 cases", the corresponding SQL query output is "select * from COVID-19". The visualization generation module is trained based on the natural language/visualization query (NL, VIS) pairs from the NL2VIS benchmark nvBench[5]. For example, the input of a natural language query is "draw a line chart to show the trend of numbers of cases by each case type in Utah", and the corresponding output is similar to Vega-Lite[2], such as "mark line encoding x date y aggregate none number color cases transform filter states = 'Utah"'.

3 An Automatic Approach for Visualization Exploration Based on MCTS

The goal of visualization exploration is to automatically generate and display appropriate visualization results based on user needs and data features. Achieving this goal faces two major challenges. The first is how to search and evaluate candidate visualization results in the vast visualization design space; the second is how to infer user preference based on the natural language queries with ambiguous user intent to generate the visualization results matching the user intent.

To address the aforementioned challenges, we propose an automatic approach for visualization exploration based on MCTS. Using the efficient search strategy of MCTS, this approach can generate and evaluate different visualization queries to recommend high-quality visualization results. A visualization query is a language used to describe visualizations, consisting of a series of visualization clauses describing different aspects of a visualization, such as data columns, chart types, and aggregation operations. In the tree T of MCTS, each node v corresponds to a visualization clause c. For instance, the nodes in the second layer in Figure 5 correspond to chart type clauses. Therefore, a visualization query Q can be represented as a path $Q = [v_1, v_2, \cdots, v_n]$ from the root node to the leaf node, where n is the number of clauses in the query.

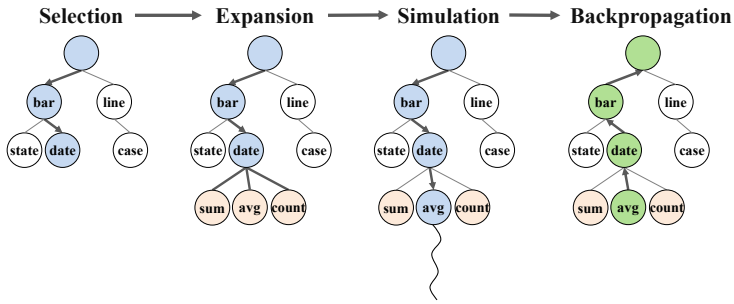


Figure 5 Four steps of Monte Carlo tree search

In response to the first challenge, we build a search tree composed of multiple visualization clauses by using dataset parsing and stepwise search methods. In the search process, a pruning algorithm and a composite reward function, both incorporating visualization domain knowledge, are devised to guide the system to select the best visualization results efficiently.

To address the second challenge, we extract user intent from input natural language queries to obtain keywords associated with the analytics tasks (such as distribution, trend, and comparison) and visualization clauses (such as column names and aggregation methods). Based on these keywords, the expansion direction of the search tree nodes can be limited, and the search tree can search according to the user preference and further prune the search space in the step of expansion. As shown in Figure 5, the MCTS includes the following four steps.

(1) Selection. In this step, the algorithm will start from the root node and recursively search for the optimal child node until it reaches a node that has not yet fully expanded. As shown in Figure 5, the blue nodes represent the nodes selected in this step. The Upper Confidence Bound (UCB) algorithm[17] is employed to fully use the feedback from the reward function and balance the relationship between exploration and exploitation, so as to avoid the locally optimal solution. The key idea of this algorithm is to select the child node with the highest upper confidence limit in each iteration for further exploration. The specific selection strategy of the UCB algorithm can be expressed as in Eq. (4):

$$\mathrm{UCB}(i) = \bar{X}_i + C\sqrt{\frac{\ln N}{n_i}} \tag{4}$$

where $\mathrm{UCB}(i)$ is the upper confidence limit of the child node i; $\bar{X}_i$ is the average reward value of the child node i; N is the number of visits to the current node; n_i is the number of visits to the child node; and C is a parameter controlling the balance between exploration and exploitation. With this selection strategy, the UCB algorithm can fully explore the node space while effectively using known information within limited times of simulations.

(2) Expansion. In this step, a pruning algorithm incorporating visualization domain knowledge is devised to narrow the search scope. This algorithm comprehensively considers information in multiple dimensions such as data type, chart type, and encoding rules and selects the candidate operations with great potential values.

(3) Simulation. In this step, the algorithm selects the current node as the starting point for simulation and expands it based on the preset constraint rules within time constraints. During the updating of simulation nodes, the algorithm prioritizes the operation with the highest potential value and computes the reward value of the simulation results to optimize the decision process.

(4) Backpropagation. In this step, the algorithm updates the number of visits and cumulative rewards of nodes along the determined path, as shown in Figure 5. The score for each

visualization result is calculated by the reward function.

Reward function design. Since visualization lacks clear reward and punishment rules like those in Go and biases may exist in the rewards based on single evaluation criteria, we comprehensively evaluate the visualization result quality considering three aspects: data features, visualization domain knowledge, and user preference.

(1) Data features. In terms of data features, we implement a scoring function through LambdaMART[18]. This method can extract useful data features (such as maximum, minimum, and fraction of unique values) and use machine learning to score the visualization.

(2) Visualization domain knowledge. Regarding visualization domain knowledge, a rule-based scoring strategy is adopted. The domain knowledge rules aim to ensure that the generated visualization results can adapt to human visualization perception while realistically presenting the core attributes of the data. For example, a bar chart composed of over 50 bars may make it difficult for users to capture important information. By incorporating domain knowledge as constraints, we can set different rules for different visualization types, thus effectively guiding the direction of exploration.

(3) User preference. The user preference is divided it into two parts: the rating in the initial stage of visualization exploration and the user rating during the interaction. In the initial stage with the lack of user preference information, a user model is trained by generative adversarial networks[19] and evaluated. The training data for this model is sourced from the visualization corpus of the plotly community forum[20], including the interaction records of real users. From these records, the model can learn the commonalities of user preferences and behavioral characteristics, and thus provide accurate initial evaluation of the visualization results. In the initial stage, the evaluation results of the model are considered as the main evaluation criteria.

During the interaction, the system records the user's interactive behavior in real time and introduces the real-time user response model[22] in the recommender system[21] to dynamically adjust user ratings. We design a reward mechanism for different user operations. When a user performs operations such as clicking, saving, or editing on a visualization result, the system will convert these operations into numerical values based on the preset rules and dynamically adjust the scores of corresponding visualization results and clauses. In this way, the system can assign higher reward scores to the results of interests to the user. In the next round of MCTS, the algorithm tends to explore the directions of interests to the user, thereby generating visualization results in line with the user's actual needs[23, 24]. The system adopts a weighted combination approach, which combines the initial rating with the current user rating and assigns a higher weight to the current user rating. This design aims to well meet the current user preference and encourage the user to conduct deeper data exploration.

Next, we introduce how to use these three parts to comprehensively evaluate the quality of a visualization result. The visualization results are preliminarily screened based on domain knowledge. The results that do not match the domain knowledge are assigned with a low score. The results that meet the domain knowledge criteria will be scored based on the data features and the data feature scores will be obtained. Subsequently, the user preference score is calculated based on the user model and the current user preference. Finally, the score of the visualization result is calculated by weighted averaging the two scores, where the weight values are set based on experience.

In summary, we propose an automatic approach for visualization exploration based on MCTS, which combines a pruning algorithm and a composite reward function, both incorporating visualization domain knowledge, to improve the result accuracy and search efficiency.

4 Validation of the Effectiveness of Navi

4.1 Quantitative experiments

4.1.1 *Effectiveness validation of the data querying module*

In this experiment we use the NL2SQL benchmark Spider[10] to evaluate the effectiveness of the data querying module. Spider is a large-scale, complex, and cross-domain dataset of NL2SQL. It contains 10,181 (NL, SQL) pairs distributed across 200 databases, covering 138 different domains. We use this experiment to validate the module effectiveness on its test set and ensures that there is no natural language query or database duplication between the training and test sets.

The performance of the module is evaluated based on two metrics[25]: exact-set-match accuracy and execution accuracy. The exact-set-match accuracy refers to whether the generated SQL query is completely consistent with the real SQL query in the standardized data structure. The execution accuracy refers to whether the results obtained from the generated SQL query after execution on the database are the same as those obtained from the actual SQL query.

As shown in Table 2, the data querying module achieves high exact-set-match accuracy and high execution accuracy, which are 71.08% and 74.37%, respectively. Compared with BRIDGE[26] and RATSQL[4], the data querying module increases the exact-set-match accuracy by 5.06% and 2.95% and the execution accuracy by 7.19% and 5.16%, respectively. This result indicates that the data querying module can more effectively convert natural language queries into correct and executable SQL queries and better adapt to databases of different domains and complexities.

Table 2 Experimental results of the data querying module (%)

Systems	Exact-set-match accuracy	Execution accuracy
BRIDGE[26]	66.02	67.18
RATSQL[4]	68.13	69.21
Data querying module	**71.08**	**74.37**

4.1.2 *Effectiveness validation of the visualization generation module*

The benchmark dataset nvBench[5] is used to validate the effectiveness of the visualization generation module. As a large, complex, and cross-domain dataset for NL2VIS tasks, nvBench consists of 750 relationship tables and 25,750 (NL, VIS) pairs, covering 105 different domains. With this experiment we validate the effectiveness of the visualization generation module on nvBench based on accuracy. Accuracy refers to whether the generated visualization query exactly matches the actual visualization query, i.e. Accuracy $= N/M$, where N is the number of matched results and M is the number of test samples.

As shown in Table 3, the accuracy of the visualization generation module reaches 76.78%, which increases by 18.13% and 12.66% compared with that of NL2Viz[27] and SEQ2VIS[5], respectively. This result indicates that the visualization generation module can flexibly handle natural language queries of different types and complexities and accurately generate visualizations that match user intent and data features. This proves that the module can effectively achieve visualization generation.

Table 3 Experimental results of the visualization generation module (%)

Systems	Accuracy
NL2Viz[27]	58.65
SEQ2VIS[5]	64.12
Visualization generation module	**76.78**

4.1.3 *Effectiveness validation of the visualization exploration module*

In this experiment we use the dataset of KG4VIS[7] to evaluate the effectiveness of the visualization exploration module. KG4VIS is a visualization exploration approach based on knowledge graphs. A total of 88,548 pairs (dataset, visualizations) are selected from VizML[28] and classified into training and test sets in a 7:3 ratio. We use the test set for evaluation and employ the evaluation metrics of KG4VIS, which include:

- average rank: the average position of the correct visualization design choices among all generated results, and a lower rank means a better result;
- Hits@2: the probability of the correct visualization design choices appearing in the first two generated results, and a higher value represents a better result;
- axis accuracy: the probability that the axis attribute (i.e. x-axis or y-axis) in the generated results is consistent with the actual result, and a higher value corresponds to a better result.

With this experiment we aim to evaluate the quality of the visualization results generated by the system for the first time. Therefore, the user preference obtained by the reward function is mainly evaluated based on the initial ratings in the visualization exploration. As shown in Table 4, the average rank, Hits@2, and axis accuracy of the visualization exploration module are 1.7268, 87.22%, and 98.12%, respectively. Compared with KG4VIS[7] and DeepEye[6], the visualization exploration module adopted in this paper decreases the average rank by 0.3414 and 0.2776, increases Hits@2 by 12.15% and 9.94%, and increases the axis accuracy by 38.11% and 4.93%, respectively, demonstrating obvious advantages. This result indicates that the visualization exploration module can generate appropriate visualizations more intelligently based on data features and user intent and more accurately determine axis attribute and data mapping. It proves that this module can effectively achieve visualization exploration.

Table 4 Experimental results of the visualization exploration module

Systems	Average rank	Hits@2 (%)	Axis accuracy (%)
KG4VIS[7]	2.0682	75.07	60.01
DeepEye[6]	2.0044	77.28	93.19
Visualization exploration module	**1.7268**	**87.22**	**98.12**

- Ablation study. The composite reward function in the visualization exploration module integrates ratings from three aspects: data features, visualization domain knowledge, and user preferences. Among them, the user preferences include two parts: the rating in the initial stage of visualization exploration and the user rating during the interaction. To validate the effectiveness of the composite reward function proposed in this study, we design five variant implementations of Navi.

(1) Navi: evaluate the quality of visualization results generated for the first time without considering the current user rating.

(2) Navi score without considering visualization domain knowledge.

(3) Navi score without considering data features.

(4) Navi score without considering user preferences.

(5) Navi+: evaluate the quality of visualization results generated by users after multiple rounds of interactions combined with the current user rating.

As shown in Table 5, the visualization results obtained when considering the user rating during the interaction have the highest quality, with Hits@2 and axis accuracy of 90.36% and 98.83%, respectively. The visualization results generated by Navi for the first time without considering the user rating showed Hits@2 and the axis accuracy decrease of 2.64% and 1.02%, respectively. The quality of the visualization results generated by Navi for the first time without considering the visualization domain knowledge, data features, or initial user preferences

Table 5 Ablation study results of the composite reward function (%)

Variant implementations of Navi	Hits@2	Axis accuracy
Navi result without considering visualization domain knowledge	78.56	92.15
Navi result without considering data features	77.82	89.31
Navi result without considering user preferences	76.78	91.22
Navi (without user interactivity)	87.72	97.81
Navi+ (with user interactivity)	**90.36**	**98.83**

declines, as manifested in the decreased Hits@2 and axis accuracy. The experimental results indicate that all the three aspects considered in the composite reward function have positive impacts on the performance of the visualization exploration module. Moreover, by recording user interactive behavior and accordingly adjusting the rating of the reward function, the quality of the generated visualization results can be improved. Therefore, the composite reward function designed in this paper can comprehensively evaluate the advantages and disadvantages of different visualization results and guide the visualization exploration module to generate high-quality visualization results.

4.1.4 *Effectiveness validation of the task scheduler*

We design this experiment to evaluate the performance of the task scheduler (described in Section 1.2) on the test set built based on Spider[10] and nvBench[5]. This experiment aims to examine whether the task scheduler can accurately recognize natural language queries input by users and assign them to appropriate functional modules. Three metrics: accuracy, recall, and $F1$ value, are used for the validation. Accuracy represents the proportion of correctly assigned queries to the corresponding module by the task scheduler. Recall represents the proportion of correctly assigned queries to the corresponding module to the total number of allocations. $F1$ value represents the harmonic mean of accuracy and recall, reflecting the balance between the two.

As shown in Table 6, the task classification model achieves the accuracy, recall, and F1 of 98.22%, 98.23%, and 98.55%, respectively, on the test set. The result indicates that the task scheduler given in this study can effectively identify different types of analytics tasks and accurately allocate them to the corresponding modules.

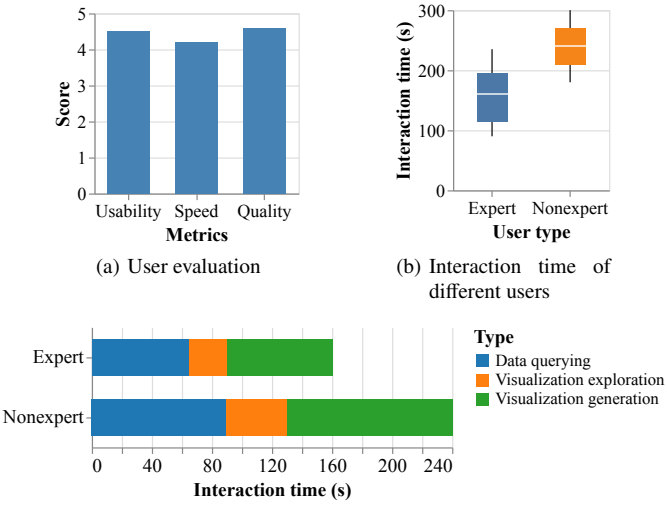
Table 6 Experimental results of the task scheduler (%)

Module	Accuracy	Recall	$F1$
Task scheduler	**98.22**	**98.23**	**98.55**

4.2 User study

Five expert users and 15 non-experts users are invited to participate in the user study and evaluate the end-to-end performance of Navi.

- User tasks. In this experiment we design different data analytics tasks for each user, including data querying, visualization generation, and visualization exploration. Users interact with the system through natural language queries and complete corresponding goals based on the results returned by the system.
- User evaluation. After completing each task, users are invited to fill out a questionnaire and rate the system in terms of usability, response speed, and result quality. Meanwhile, the time and operation logs taken by users to complete each task are recorded.
- Study results. Figure 6(a) illustrates the user satisfaction for Navi in various evaluation dimensions, with an average score exceeding 4 out of 5. This result fully reflects the outstanding performance of Navi in data analytics, which provides a convenient and efficient user experience. Figure 6(b) shows the interaction time of different types of



(a) User evaluation

(b) Interaction time of different users

(c) Interaction time in different operations

Figure 6 User study results

users when using Navi. The interaction time of some non-expert users is close to that of expert users, suggesting that Navi can effectively improve the data analytics capability and efficiency of users. As shown in Figure 6(c), the average interaction time of users is the shortest in the visualization exploration, which not only highlights the high efficiency of the visualization exploration module but also demonstrates the effectiveness of Navi in helping users quickly probe into data insights.

5 Related Work

NL2SQL. In the era of big data, it is of great significance to efficiently retrieve useful information from vast datasets[29, 30]. NL2SQL emerging as the times require allows users to efficiently retrieve information from structured data through natural language queries. The recent years have witnessed significant research progress in the NL2SQL models based on deep learning in the fields of databases and natural language processing[31–36]. Representative NL2SQL models such as RESDSQL[32], RASAT[33], and Graphix-T5[34] are trained by the fine-tuning-based methods. These models require a large number of (NL, SQL) pairs for training and evaluated for the performance on benchmark datasets such as Spider[10]. As Large Language Models (LLMs) are emerging, researchers have begun to explore the potential applications of these models in NL2SQL tasks. Models such as C3[35] and DIN-SQL[36] have demonstrated satisfactory performance in NL2SQL tasks through zero-shot and few-shot prompts. However, the available LLMs face challenges in handling relationships between multiple tables and columns in databases, demonstrating inferior performance to conventional fine-tuning-based methods when dealing with complex query tasks[32].

NL2VIS. NL2VIS is mainly implemented based on the rule-based methods and deep learning methods[37, 38]. The representative rule-based methods include NL4DV[39], QRec-NLI[40], and NL2Viz[27]. These methods mainly parse natural language queries by semantic parsers such as NLTK[41], Stanford CoreNLP[42], and NER[43], extract language features such as part of speech, named entities, and coreference, and ultimately convert them into corresponding visualization queries through heuristic rules. However, these methods have limitations in dealing

with the fuzziness of natural language, with the robustness remaining to be improved.

In recent years, the breakthrough progress of deep learning in natural language processing[37] has provided a new technological path for NL2VIS. Researchers have introduced deep learning into NL2VIS tasks, among which the representative research achievements include ADVISor[44], ncNet[45], and Chat2vis[46]. ADVISor adopts a BERT-based framework[47], which first completes the mapping of NL2SQL and then generates visualization queries based on rules. Being an end-to-end model based on Transformer[12], ncNet uses nvBench[5], the first common benchmark of NL2VIS, for training. This dataset includes (NL, VIS) pairs from various domains and scenarios, providing a criterion for training and evaluating deep learning models. Chat2vis combines LLMs (such as Codex[48] and GPT-3[49]) with prompt engineering[50] to achieve the transformation from natural language queries to visualization queries, promoting the research progress in NL2VIS.

Visualization exploration. Creating visualization results that align with user analytics intent is a challenging task. It requires users to understand the data, have clear analytics intent, and be familiar with visualization techniques. Therefore, it is particularly important to develop data analysis systems that can automatically conduct visualization exploration. For example, Draco[51] and DeepEye[6, 52, 53] can provide ranked visualization results for specific datasets. Draco combines hard and soft constraints to identify and rank high-quality visualization results. DeepEye, on the other hand, uses a decision-tree model to evaluate the quality of visualizations and uses learning-to-rank[54] for visualization ranking. To make the visualization exploration process more relevant to the user intent, more advanced technologies are introduced. For instance, LineNet[55] adopts triplet autoencoder based on vision transformer[56] to automatically recommend similar line charts based on the line charts provided by users. Lux[57] is a tool that can be integrated into Jupyter Notebook, providing real-time data analytics support based on the user's actual analytics intent. Sevi[58] employs voice interaction to obtain user analytics intent, facilitating the visualization generation and exploration through voice interaction.

6 Conclusion and Future Work

This study designs Navi, a data analysis system powered by natural language interaction. Following the modular design philosophy, Navi integrates three functional modules: data querying, visualization generation, and visualization exploration, and is managed uniformly by a task scheduler. Quantitative experiments and user study have shown that Navi can effectively support users in data analytics operations. The future work can be carried out to support multiple rounds of dialogue and integrate contextual information, thus enabling users to continuously perform data analytics operations. Secondly, efforts can be made to explore the integration with LLMs, the advantages of which in natural language processing can be leveraged to enhance the performance and effectiveness of Navi in diverse tasks.

References

[1] Ward MO, Grinstein G, Keim D. Interactive Data Visualization: Foundations, Techniques, and Applications. CRC Press, 2010.

[2] Satyanarayan A, Moritz D, Wongsuphasawat K, Heer J. Vega-Lite: A grammar of interactive graphics. IEEE Trans. on Visualization and Computer Graphics, 2016, 23(1): 341–350. [doi: 10.1109/TVCG.2016.2599030]

[3] Tableau. 2023. https://www.tableau.com/.

[4] Shi P, Ng P, Wang Z, Zhu HH, Li AHB, Wang J, dos Santos CN, Xiang B. Learning contextual representations for semantic parsing with generation-augmented pre-training. Proc. of the AAAI Conf. on Artificial Intelligence, 2021, 35(15): 13806–13814. [doi: 10.1609/aaai.v35i15.17627]

[5] Luo Y, Tang N, Li G, Chai CL, Li WB, Qin XD. Synthesizing natural language to visualization

(NL2VIS) benchmarks from NL2SQL benchmarks. Proc. of the 2021 Int'l Conf. on Management of Data. 2021. 1235–1247. [doi: 10.1145/3448016.3457261]

[6] Luo Y, Qin X, Chai C, Tang N, Li GL, Li WB. Steerable self-driving data visualization. IEEE Trans. on Knowledge and Data Engineering, 2020, 34(1): 475–490. [doi: 10.1109/TKDE.2020.2981464]

[7] Li H, Wang Y, Zhang S, Song YQ, Qu HM. KG4Vis: A knowledge graph-based approach for visualization recommendation. IEEE Trans. on Visualization and Computer Graphics, 2021, 28(1): 195–205. [doi: 10.1109/TVCG.2021.3114863]

[8] Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S. A survey of Monte Carlo tree search methods. IEEE Trans. on Computational Intelligence and AI in Games, 2012, 4(1): 1–43. [doi: 10.1109/TCIAIG.2012.2186810]

[9] Munzner T. Visualization Analysis and Design. CRC Press, 2014.

[10] Yu T, Zhang R, Yang K, Yasunaga M, Wang DX, Li ZF, Ma J, Li I, Yao QN, Roman S, Zhang ZL, Radev D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. arXiv:1809.08887, 2018.

[11] Sanh V, Debut L, Chaumond J, Wolf T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv:1910.01108, 2019.

[12] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Advances in Neural Information Processing Systems, Vol. 30. 2017. 5998–6008.

[13] Loshchilov I, Hutter F. Decoupled weight decay regularization. arXiv:1711.05101, 2017.

[14] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, Vol. 27. 2014. 3104–3112.

[15] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP). 2014. 1532–1543.

[16] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014.

[17] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem. Machine Learning, 2002, 47: 235–256.

[18] Wu Q, Burges CJC, Svore KM, Gao JF. Ranking, boosting, and model adaptation. Technical Report, MSR-TR-2008-109, Microsoft Research, 2008.

[19] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial networks. Communications of the ACM, 2020, 63(11): 139–144. [doi: 10.1145/3422622]

[20] Qian X, Rossi RA, Du F, Kim S, Koh E, Malik S, Lee KY, Ahmed NK. Personalized visualization recommendation. ACM Trans. on the Web (TWEB), 2022, 16(3): 1–47. [doi: 10.1145/3538703]

[21] Afsar MM, Crump T, Far B. Reinforcement learning based recommender systems: A survey. ACM Computing Surveys, 2022, 55(7): 1–38. [doi: 10.1145/3543846]

[22] Chen M, Chang B, Xu C, Chi ED. User response models to improve a reinforce recommender system. Proc. of the 14th ACM Int'l Conf. on Web Search and Data Mining. 2021. 121–129. [doi: 10.1145/3437963.3441764]

[23] Baghi V, Motehayeri SMS, Moeini A, Abedian R. Improving ranking function and diversification in interactive recommendation systems based on deep reinforcement learning. Proc. of the 26th Int'l Computer Conf., Computer Society of Iran (CSICC). IEEE, 2021. 1–7. [doi: 10.1109/CSICC52343.2021.9420615]

[24] Xiao T, Wang D. A general offline reinforcement learning framework for interactive recommendation. Proc. of the AAAI Conf. on Artificial Intelligence, 2021, 35(5): 4512–4520. [doi: 10.1609/aaai.v35i5.16579]

[25] Zhong RQ, Yu T, Klein D. Semantic evaluation for text-to-SQL with distilled test suites. Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). 2020. 396–411.

[26] Lin XV, Socher R, Xiong C. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. arXiv:2012.12627, 2020.

[27] Wu Z, Le V, Tiwari A, Gulwani S, Radhakrishna A, Radiček I, Soares G, Wang XY, Li ZW, Xie T. NL2Viz: Natural language to visualization via constrained syntax-guided synthesis. In: Proc. of the 30th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. 2022. 972–983. [doi: 10.1145/3540250.3549140]

[28] Hu K, Bakker MA, Li S, Kraska T, Hidalgo C. Vizml: A machine learning approach to visualization recommendation. Proc. of the 2019 CHI Conf. on Human Factors in Computing Systems. 2019. 1–12. [doi: 10.1145/3290605.3300358]

[29] Qin XD, Chai CL, Luo YY, Zhao TY, Tang N, Li GL, Feng JH, Yu X, Ouzzani M. Interactively discovering and ranking desired tuples by data exploration. The VLDB Journal, 2022, 31(4): 753–777. [doi: 10.1007/s00778-021-00714-0]

[30] Chai C, Liu J, Tang N, Fan J, Miao DJ, Wang JY, Luo YY, Li GL. GoodCore: Data-effective and data-efficient machine learning through coreset selection over incomplete data. Proc. of the ACM on Management of Data, 2023, 1(2): 1–27. [doi: 10.1145/3589302]

[31] Kim H, So BH, Han WS, Lee H. Natural language to SQL: Where are we today? Proc. of the VLDB Endowment, 2020, 13(10): 1737–1750. [doi: 10.14778/3401960.3401970]

[32] Li HY, Zhang J, Li CP, Chen H. RESDSQL: Decoupling schema linking and skeleton parsing for text-to-SQL. Proc. of the AAAI Conf. on Artificial Intelligence, 2023, 37(11): 13067–13075. [doi: 10.1609/aaai.v37i11.26535]

[33] Qi J, Tang J, He Z, Wan XP, Cheng Y, Zhou CH, Wang XB, Zhang QS, Lin ZH. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing. 2022. 3215–3229.

[34] Li J, Hui B, Cheng R, Qin B, Ma CH, Huo N, Huang F, Du WY, Si L, Li YB. Graphix-T5: Mixing pre-trained Transformers with graph-aware layers for text-to-SQL parsing. Proc. of the 2023 AAAI. 2023. 13076–13084.

[35] Dong X, Zhang C, Ge Y, Mao YR, Gao YJ, Chen L, Lin JS, Lou DF. C3: Zero-shot text-to-SQL with ChatGPT. arXiv:2307.07306, 2023.

[36] Pourreza M, Rafiei D. Din-SQL: Decomposed in-context learning of text-to-SQL with self-correction. arXiv:2304.11015, 2023.

[37] Shen L, Shen E, Luo Y, Yang XC, Hu XM, Zhang XS, Tai ZW, Wang JM. Towards natural language interfaces for data visualization: A survey. IEEE Trans. on Visualization and Computer Graphics, 2023, 29(6): 3121–3144. [doi: 10.1109/TVCG.2022.3148007]

[38] Luo YY, Qin XD, Xie YP, Li GL. Intelligent data visualization analysis techniques: A survey. Ruan Jian Xue Bao/Journal of Software, 2024, 35(1): 356–404 (in Chinese with English abstract). https://www.jos.org.cn/1000-9825/6911.htm [doi: 10.13328/j.cnki.jos.006911]

[39] Narechania A, Srinivasan A, Stasko J. NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. IEEE Trans. on Visualization and Computer Graphics, 2020, 27(2): 369–379. [DOI: 10.1109/TVCG.2020.3030378]

[40] Wang X, Cheng F, Wang Y, Xu K, Long J, Lu H, Qu HM. Interactive data analysis with next-step natural language query recommendation. arXiv:2201. 04868, 2022.

[41] Bird S. NLTK: The natural language toolkit. Proc. of the COLING/ACL 2006 Interactive Presentation Sessions. 2006. 69–72.

[42] Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D. The Stanford CoreNLP natural language processing toolkit. Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2014. 55–60.

[43] Finkel JR, Grenager T, Manning CD. Incorporating non-local information into information extraction systems by Gibbs sampling. Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005). 2005. 363–370.

[44] Liu C, Han Y, Jiang R, Yuan XR. Advisor: Automatic visualization answer for natural-language question on tabular data. Proc. of the 14th IEEE Pacific Visualization Symp. (PacificVis). IEEE, 2021. 111120. [doi: 10.1109/PacificVis52677.2021.00010]

[45] Luo YY, Tang N, Li GL, Tang JW, Chai CL, Qin XD. Natural language to visualization by neural

machine translation. IEEE Trans. on Visualization and Computer Graphics, 2021, 28(1): 217–226. [doi: 10.1109/TVCG.2021.3114848]

[46] Maddigan P, Susnjak T. Chat2vis: Generating data visualisations via natural language using ChatGPT, codex and GPT-3 large language models. arXiv:2302.02094v2, 2023.

[47] Devlin J, Chang MW, Lee K, Toutanova L. BERT: Pre-training of deep bidirectional Transformers for language understanding. Proc. of the 2019 NAACL-HLT. 2019. 4171–4186.

[48] Chen M, Tworek J, Jun H, *et al*. Evaluating large language models trained on code. arXiv:2107.03374, 2021.

[49] Brown T, Mann B, Ryder N, *et al*. Language models are few-shot learners. arXiv:2005.14165, 2020.

[50] Zhou YC, Muresanu AI, Han ZW, Paster K, Pitis S, Chan H, Ba J. Large language models are human-level prompt engineers. arXiv:2211.01910, 2022.

[51] Moritz D, Wang CL, Nelson GL, Lin H, Smith AM, Howe B, Heer J. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. IEEE Trans. on Visualization and Computer Graphics, 2019, 25(1): 438–448. [doi: 10.1109/TVCG.2018.2865240]

[52] Luo YY, Qin XD, Tang N, Li GL. DeepEye: Towards automatic data visualization. Proc. of the 34th IEEE Int'l Conf. on Data Engineering (ICDE). IEEE, 2018. 101–112. [doi: 10.1109/ICDE.2018.00019]

[53] Luo Y, Qin X, Tang N, Li GL, Wang XR. DeepEye: Creating good data visualizations by keyword search. Proc. of the 2018 Int'l Conf. on Management of Data. 2018. 1733–1736. [doi: 10.1145/3183713.3193545]

[54] Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G. Learning to rank using gradient descent. Proc. of the 22nd Int'l Conf. on Machine Learning. 2005. 89–96. [doi: 0.1145/1102351.1102363]

[55] Luo YY, Zhou YH, Tang N, Li GL, Chai CL, Shen LX. Learned data-aware image representations of line charts for similarity search. Proc. of the ACM on Management of Data, 2023, 1(1): 1–29. [doi: 10.1145/3588942]

[56] Liu Z, Lin Y, Cao Y, Hu H, Wei YX, Zhang Z, Lin S, Guo BN. Swin transformer: Hierarchical vision transformer using shifted windows. Proc. of the IEEE/CVF Int'l Conf. on Computer Vision. 2021. 10012–10022.

[57] Lee DJL, Tang DX, Agarwal K, Boonmark T, Chen C, Kang J, Mukhopadhyay U, Song J, Yong M, Hearst MA, Parameswaran AG. Lux: Always-on visualization recommendations for exploratory dataframe workflows. Proc. of the VLDB Endowment, 2021, 15(3): 727–738.

[58] Tang J, Luo Y, Ouzzani M, Li GL, Chen HY. Sevi: Speech-to-visualization through neural machine translation. Proc. of the 2022 Int'l Conf. on Management of Data. 2022. 2353–2356. [doi: 10.1145/3514221.3520150]



Yupeng Xie, master's degree candidate. His main research interest is data visualization.



Jianhua Feng, Ph.D., professor, doctoral supervisor. His main research interests include databases, data security and privacy protection, and information retrieval.



Yuyu Luo, Ph.D., associate researcher, doctoral supervisor. His main research interests include intelligent data management and visualization analytics.