Visualization Recommendation with Prompt-based Reprogramming of Large Language Models

Anonymous ACL submission

Abstract

Visualization recommendations, which aim to automatically match proper visual charts for specific data tables, can significantly simplify the data analysis process. Traditional approaches in this domain have primarily relied on rule-based or machine learning-based methodologies. These methods often demand extensive manual maintenance and yet fail to fully comprehend the tabular data, leading to unsatisfactory performance. Recently, Large Language Models (LLMs) have emerged as powerful tools, exhibiting strong reasoning capabilities. This advancement suggests their substantial promise in addressing visualization recommendation challenges. However, effectively harnessing LLMs to discern and rationalize patterns in tabular data, and consequently deduce the essential information for chart generation, remains an unresolved challenge. To this end, we introduce a novel Hierarchical Table Prompt-based reprogramming framework, named HTP. This framework aims to integrate multi-dimensional tabular data into LLMs through a strategically crafted prompt learning method while keeping the LLMs' backbone and weights unaltered. The HTP framework uniquely incorporates a four-level prompt structure, encompassing *general, instance, cluster*, and *column* levels. This multi-level approach is engineered to provide a comprehensive understanding of both general distribution and multifaceted fine-grained features of tabular data, before inputting the tabular data into the frozen LLM. Our empirical studies confirm that the HTP framework achieves state-of-the-art performance, marking an advancement in the field of data visualization and analysis. The code and data will be made publicly available upon acceptance.

1 Introduction

Data visualization, which facilitates effective decision-making via better responding the human

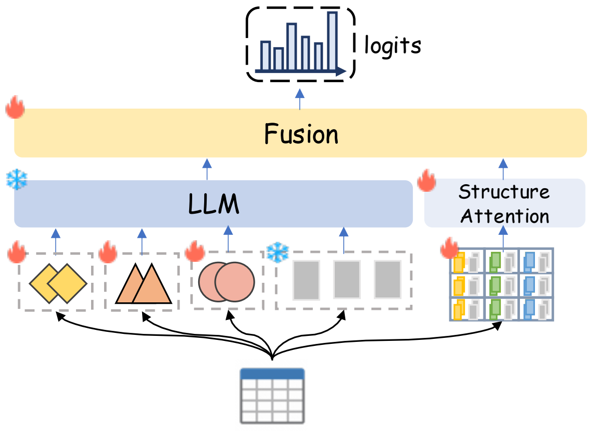

Figure 1: Illustration of HTP framework.

sensitivity and efficiency in processing visual information (Kosmyna et al., 2018), has become increasingly crucial nowadays. Traditional visualization tools necessitate extensive effort for manual specification, which often requires domain expertise in data analysis. Therefore, large efforts have been made to develop automatic tools for solving the *visualization recommendation* task, i.e., recommending the proper visual charts.

Generally, prior arts can be divided into two categories. The first, known as *Rule-based* approaches (Mackinlay, 1986; Mackinlay et al., 2007), relies on expert-designed rules to generate visualizations. However, these approaches not only suffer the massive manual labor for rule maintenance, but also struggle to address the combinatorial explosion problem caused by increasing data dimensions. In contrast, the *Machine Learning-based* approaches (Dibia and Demiralp, 2019; Li et al., 2021) are investigated to automatically learn the best matching between tabular data and visualization elements. Unfortunately, few of them could fully extract the multi-level features within tables, which severely limits the performance.

Last year has witnessed the prosperity of Large Language Models (LLMs), whose advanced capabilities of LLM in processing and interpreting complex data unlocks new possibilities in the field of data visualization. However, significant challenges persist in effectively harnessing the potential of an LLM for visualization recommendation. Firstly, LLMs are typically designed to process sequences of discrete tokens in an unstructured format, whereas tables used for visualization often display highly structured and predominantly numerical characteristics, which leads to a mismatch. Second, the inherent pre-training of LLMs does not naturally encompass the comprehension abilities required to interpret complex tabular data. Third, rather than fine-tuning the entire model, it is often preferable to keep the LLM backbone "frozen". This approach ensures the model's versatility in supporting a variety of tasks without compromising its fundamental capabilities.

To that end, we propose a novel **H**ierarchical **T**able **P**rompt-based reprogramming framework (HTP), to adapt LLM for visualization recommendation without altering the backbone structure. The essence of HTP lies in leveraging data-driven multi-level prompts to adaptively reprogram the tabular data across various dimensions, thereby bridging the gap between the structured nature of tabular data, as well as the inherent capabilities of LLMs for textual processing. Specifically, HTP utilizes four levels of prompts as follows:

- **General-level prompt**, which is employed to describe the overall distribution of table dataset, while facilitating information sharing and integration among prompts at various levels, thereby enhancing the generalization performance of LLM.
- **Instance-level prompt**, which connects the individual table instances with various charts, by leveraging the specific distribution of tabular data and corresponding chart.
- **Cluster-level prompt**, which is generated via feature extraction and clustering, and targets enhancing the LLM to capture the implicit patterns that exist within the table datasets, as well as the correlations among patterns.
- **Column-level prompt**, which originates from the inherent structural information of tables, and highlights the columnar organization to improve the column-level information processing, and support cross-column comprehension.

Based on these four prompts, as illustrated in Figure 1, we concatenate prompts at *general, instance*, and *cluster* levels with a serialized table before feeding them into LLMs, thereby facilitating table-specific knowledge extraction and integration. Meanwhile, the *column-level* prompts are prepended to encoded inputs to retain the structural information of tables, thereby improving the distinction among columns, and fostering a contextual understanding between them. The contribution of this paper could be summarized as follows:

- To the best of our knowledge, we are the first to leverage LLMs to investigate the visualization recommendation task without altering the pre-trained backbone model.
- A novel prompt-based framework is proposed to reprogram the hierarchical table information into multi-prompts, which enhances the comprehension capabilities of LLMs.
- Extensive experiments on real-world datasets demonstrate the effectiveness of the proposed HTP compared with state-of-the-art approaches.

2 Related Work

2.1 Visualization Recommendation

Prior researches on automated visualization recommendation includes rule-based and machine learning-based approaches. Rule-based approaches (Mackinlay, 1986; Roth et al., 1994; Perry et al., 2013; Wongsuphasawat et al., 2016) are based on manually developed rules, which heavily rely on expert knowledge, and have high maintenance costs. Moreover, as table size grows, these methods may encounter combinatorial explosion issues. Recently, machine learning-based methods have shown notable progress in enhancing recommendation accuracy and scalability. (Luo et al., 2018) employed expert rules for initial visualizations, and evaluated them using a trained classifier. (Li et al., 2021; Hu et al., 2019) formulated the visualization recommendation challenge as a series of classification tasks, learn to predict labels for various design choices, while (Zhou et al., 2020; Dibia and Demiralp, 2019) cast the challenge as a sequence generation problem. The former utilized a Deep Q-Network (DQN) for selecting action tokens to fill the predefined chart templates, while the latter employed a seq2seq model to autoregressively generate JSON-encoded charts. However, these approaches are limited to learning the existing knowledge within datasets, and lack sufficient exploration of multifaceted table features, such as table patterns and structures, leading to a limited comprehension of complicated tables. Different from prior arts, in our work, we enrich the extensive pre-trained knowledge of LLMs with multi-level prompt, to fully extract the semantics within table data from multiple dimensions.

2.2 Prompt Tuning

With continuous parameter scaling of pre-training models, fine-tuning entire models for downstream tasks becomes daunting due to inefficiency and potential catastrophic forgetting (Pfeiffer et al., 2021; Kan et al., 2023). Thus, Prompt Tuning (Lester et al., 2021), which prepends learnable continuous prompts to input text while keeping model parameters fixed, has achieved success across various domains. However, directly using this method to complex table datasets results in substantial information loss. In this paper, we use multi-level prompts to integrate table information, enhancing the efficiency of utilizing prior knowledge.

3 Preliminaries

3.1 Problem Formulation

We formulate the visualization recommendation task as a text generation problem. Consider a labeled training dataset $\mathcal{D} = \{(T, V)_i\}_{i=1}^{|\mathcal{D}|}$ where T_i represents a serialized table and $V_i = v_i^1, v_i^2, \cdots v_i^{l_i}$ is a concise json description of chart with length l_i, which encapsulating key information for defining a chart. Our goal is to generate V_i in an auto-regressive way. Based on Prompt Tuning (Lester et al., 2021), we learn a prompts group $\mathcal{P}$ that have multi-level prompts, with a module $\mathcal{G}$ designed to dynamically produce prompts $\mathcal{P}$. We train our model by maximizing the likelihood of generating the target sequence, as follows:

$$\max_{\mathcal{P},\mathcal{G}} p_\theta \prod_{i=1}^{|\mathcal{D}|} \prod_{j=1}^{l_i} (v_i^j \mid v_i^1, v_i^2 \cdots v_i^{j-1}, T_i, \mathcal{P}, \mathcal{G}). \tag{1}$$

Here parameters θ of LLM remain frozen, only the prompts group $\mathcal{P}$ and $\mathcal{G}$ are updated.

3.2 Data Serialization

3.2.1 Table Linearization

As demonstrated in (Suadaa et al., 2021; Zhang et al., 2020), the representation of data in table form has a significant impact on generation performance. In this paper, we adopt a template-based linearization approach, transforming tables into flat string representations to enhance their compatibility with the internal structures of LLMs. Formally, given a table T, it is represented as:

$< header_{1,1} >$ *is* $< value_{1,1} >, \cdots < header_{i,j} >$ *is* $< value_{i,j} >, \cdots$.

3.2.2 Chart Mapping

We employs a JSON-based template to encode chart data similar to (Poco and Heer, 2017; Satyanarayan et al., 2017). The template encapsulates the key information necessary for defining a chart, including the visualization types and the arrangement of x-/y-axes. The details of our template are as follows:

```
{'encoding':
    {
        'x': {'field':<xfield>},
        'y': [
            {'field': <yfield1>, 'type': <trace1_type>},
            {'field': <yfield2>, 'type': <trace2_type>},
            ...
            ]
    }
}
```

where each object in "y" and the objects of "x" form a visual trace.

4 Methodology

In this section, we will introduce the proposed framework in detail. As depicted in Figure 2, our framework consists of four primary prompts, namely *general, instance, column* and *cluster* level prompts, to inject table insights into the LLM. Technical details of each component are discussed in the following subsections, step-by-step.

4.1 General-level Prompt Generation

To encompass the overall information of the table dataset and enhance generalization ability, we introduce a soft prompt P_{gen} as (Lester et al., 2021) do, which is uniformly applied across all instances. This general prompt ensures that the model comprehensively understanding table dataset. It also facilitates sharing and integrating information from other level prompts.

4.2 Instance-level Prompt Synthesization

Since tables target for visualizing data across diverse domains, there are inherent differences among tables within the dataset. Obviously, a single general prompt is insufficient to fully adapt to these complicated variations of tables. In response, we introduce an instance-aware prompt capable of dynamically capturing the table-specific features. In detail, we first synthesize a style vector that reflects the characteristics of different types of charts, named style query. Then, we use this

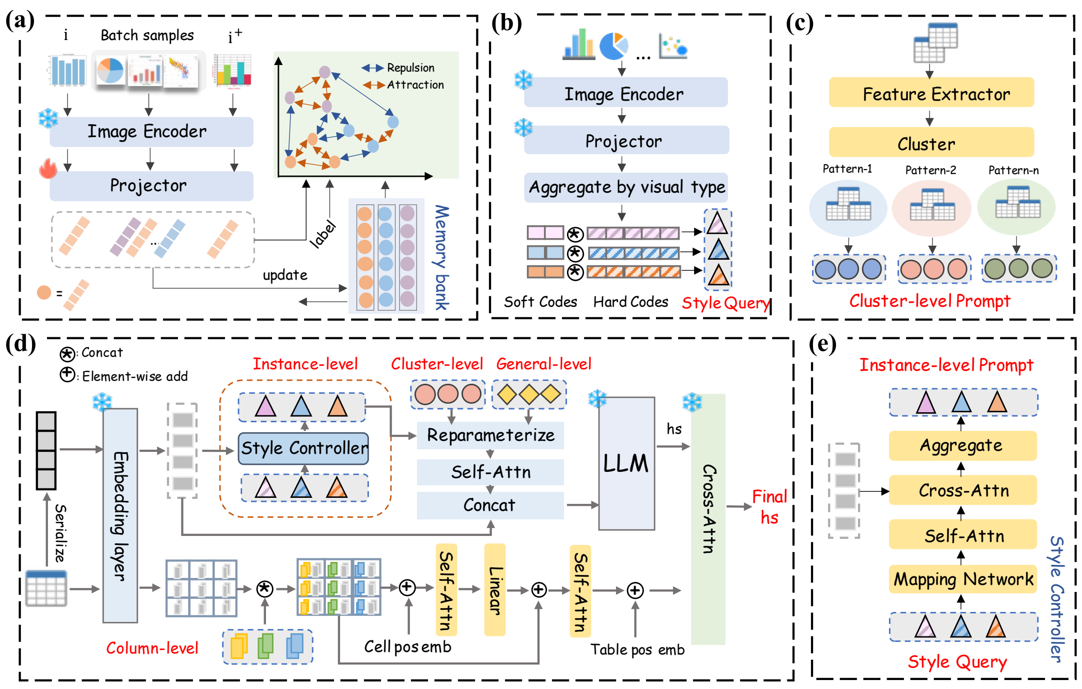

Figure 2: The overall framework of HTP. (a) Supervised contrastive learning for chart style representations. (b) Style query generation. (c) Clustering table features to generate cluster-level prompts. (d) The overall workflow of the model. (e) The detailed structure of Style Controller.

style query to project information from the serialized table embedding into the chart space. In this way, the style query facilitates the extraction of fine-grained information from tables across various dimensions, enhancing the table's perceptibility to charts. Moreover, projecting table information into chart space narrows down the search space for the LLM, thereby reducing the likelihood of producing hallucinations.

4.2.1 Style Query Generation

To ensure that the style query adequately represents the stylistic characteristics of each chart type, our style query is composed of two parts: hard codes and soft codes. Hard codes capture the inter-class features among different chart types, while soft codes are employed to explore the intra-class features within each chart type.

Hard Codes Generation. Since chart images have explicit labels (e.g., bar charts, pie charts), enabling distinction of different chart styles and further guiding chart generation. Thus, we employ a supervised contrastive learning approach that trains a projector $\mathcal{P}$ on labeled chart images, enabling it to learn the distinct representation of each chart style.

The detailed process is shown in Figure 2 (a). During training, we first feed a chart image i alongside its augmented version i^+ (e.g., rotation, grayscale), as well as other batch samples, into a frozen pre-trained image encoder to obtain visual embeddings. These extracted embeddings are then fed into a bottleneck architecture projector $\mathcal{P}$, which maps them to vectors that reflecting chart style.

Then, we utilize supervised contrastive learning loss on generated vectors to train the projector $\mathcal{P}$. Drawing inspiration from MOCO (He et al., 2020), we maintain a continuously updated memory bank for each chart type. In our work, positive samples for a given instance include its augmented versions, and same-type instances within batch and memory bank. Correspondingly, negative samples encompass instances of different chart types within the same batch and memory bank, which ensures that the model learns to distinguish different chart categories.

As shown in Figure 2 (b), after the contrastive learning phase, we reprocess the visual embeddings of images through projector $\mathcal{P}$ to obtain style features. To obtain an overall stylistic representation for each chart type, we categorize these features by chart type and perform an averaging operation within each category. In this way, we obtain a stylistic representation set $\{q_h^i\}_{i=1}^m$, named hard codes, where m represents the number of chart types.

Soft Codes Generation. Even within the same chart category, there are many subtle variations.

Relying solely on hard codes could lead to overlooking significant intra-class differences among chart types, limiting the model's ability to detect subtle stylistic differences between charts. To more effectively explore the intra-class features among charts, we introduce m trainable soft style codes $\{q_s^i\}_{i=1}^m$, which are prepended to the hard codes to provide a more comprehensive representation of the chart space.

Now the style query can be defined as follows:

$$Q_s = \{[q_s^i; q_h^i]\} \quad i = 1, 2, ..., m.$$

Here $[\cdot;\cdot]$ denotes the concatenation operation. Consequently, the style query Q_s captures intra and inter-class characteristics of each chart type simultaneously, which will be updated during training, allowing a better perception of input table.

4.2.2 Generating Instance-level Prompts with Style Query

We then use style query to guide instance-level prompt generation through a structure called Style Controller. As depicted in Figure 2 (e), we first employ a mapping network $\mathcal{M}$ to align dimension and modality between Q_s and the serialized input embedding T_r. For efficiency, $\mathcal{M}$ consists of down and up projection layers, with a nonlinear layer situated between them. Then we adopt two attention layers: the self-attention over Q_s, followed by cross-attention from T_r, as follows:

$$\begin{aligned} Q_s &= \mathcal{M}(Q_s), \\ Q_s &= Self\text{-}Attn(Q_s), \\ S &= Cross\text{-}Attn(Q_s, T_r). \end{aligned} \tag{2}$$

The $S = \{s_1, s_2, ...s_m\}$ represents features extracted by style query from m spaces based on the distributional information of table embedding. Intuitively, self-attention layer enables the model to effectively process and integrate contextual information from style query. While the cross-attention layer can dynamically emphasize the essential chart-related information within a table through style query. Finally, we generate the instance-level prompt P_{ins} by averaging the $s_i \in S$.

4.3 Cluster-level Prompt Generation

When two tables exhibit similar messages, they are likely to be represented by similar charts. Thus, using the shared information within table pattern is crucial for comprehending table representations and making chart recommendations. The diverse features of table data, such as trends, entropy, and variance, provide a comprehensive overview of table data, thus play a crucial role in identifying distinct table patterns. Thus, we extract various table features and cluster them into k different table patterns. Each table schema has its own shared prompt, resulting in a cluster prompts pool $\{p_c^i\}_{i=1}^k$. Correspondingly, for a given table $T \in C_i$, its cluster prompt can be directly set as p_c^i.

4.4 Column-level Prompt Generation

The previous discussion emphasized the overall semantic understanding of the table. However, since the selection of chart type and the arrangement of x-/y-axes are fundamentally based on column data, a detailed understanding of each column's role and characteristics within the table is also critical for recommending appropriate charts. To this end, we introduce column-level prompt that designed to encapsulate column structure information within table, facilitating intra-column understanding and cross-column analysis.

As shown in Figure 2 (d), following (Chen et al., 2022), we first encode a given table T on a per-cell basis to retain structure. Specifically, for each cell in T, we concatenate the cell's header and value, and query the LLM to obtain embeddings. To enhance representation at the column level, we introduce a unique soft prompt for each column, which is prepended to the embeddings of cells within the same column, yielding a refined, column-centric table structure representation $E \in \mathbb{R}^{m \times n \times s \times d}$, with s, m and n denoting sequence length, number of rows and columns, respectively.

Next, we employ a two-layer attention structure, the first is a cell-wise layer that focus on individual cells, while the second operates on single column to explore column semantics:

$$\begin{aligned} E_0 &= E + E_{cpe}, \\ \hat{E}_0 &= Linear(Self\text{-}Attn(E_0)) + E_0, \\ E_1 &= \frac{1}{s} \sum\nolimits_{i=1}^{s} \hat{E}_0[:, :, i, :], \\ \hat{E}_1 &= Self\text{-}Attn(E_1), \end{aligned} \tag{3}$$

where $E_{cpe} \in \mathbb{R}^{s \times d}$ is the cell text position embedding, which will be updated during training. In this way, the $\hat{E}_1$ can grasp the column structure information inside the table.

4.5 Prompt Integration for Reprogramming LLM

After obtaining the general, instance, and cluster level prompts, we use a shallow network $\mathcal{V}$ with a skip connection to reparameterize them. Subsequently, the reparameterized prompts are concatenated:

$$\begin{aligned} P' &= [\mathcal{V}(P_{gen}); \mathcal{V}(P_{ins}); \mathcal{V}(P_{clu})], \\ \mathcal{V}(P) &= \phi(P) + P, \end{aligned} \tag{4}$$

where ϕ is a simple linear layer. Furthermore, a self attention layer is applied to enable information sharing among the prompts:

$$P_{pre} = Self\text{-}Attn(P'). \tag{5}$$

The refined prompts $P_{pre} \in \mathbb{R}^{l_p \times d}$ are concatenated with the serialized table embedding $T_r \in \mathbb{R}^{l_s \times d}$, where l_p denotes the prefix length, l_s denotes the input sequence length. They are then fed into the LLM to obtain the last hidden states H. After that, we apply a cross-attention layer, using H as a query, to integrate the structured and unstructured information from the entire table:

$$\begin{aligned} E_2 &= \hat{E}_1 + E_{tpe}, \\ H &= Cross\text{-}Attn(H, E_2) + H, \end{aligned} \tag{6}$$

where $E_{tpe} \in \mathbb{R}^{m \times n \times d}$ is the table position embedding that provides row and column information for tables. This obtained hidden state will replace the original one to estimate probability of next word.

5 Experiments

5.1 Settings

Datasets Due to inaccessibility of the Plotly Corpus, a publicly available dataset collected by (Hu et al., 2019), we independently re-extracted a substantial dataset of $443,526$ public visualization pairs from the Plotly community feed via the Plotly REST API[1]. Following rigorous data cleaning, the refined dataset contains $116,528$ visualization pairs across $439,001$ columns, including $4,091(3.51\%)$ pie, $36,576(31.38\%)$ line, $29,740(25.52\%)$ scatter, $6,269(5.38\%)$ histogram, $29,641(25.43\%)$ bar and $10,211(8.77\%)$ box charts. More details are in Appendix A.

Baselines To evaluate the performance of our HTP framework, we compare our method with following baselines: (1) VizML (Hu et al., 2019), (2) KG4Vis (Li et al., 2021), (3) MultiVision (Wu et al., 2021), (4) Data2Vis (Dibia and Demiralp, 2019), (5) Table2Charts (Zhou et al., 2020), (6) DeepEye (Luo et al., 2018). More details are in Appendix B.

[1]https://api.plot.ly/v2/

Metrics Referring to VizML, we employ Accuracy to evaluate whether models can recommend correct design choices, including: XY (whether the field encoded on the X-axis and Y-axis is correct), chart type (whether the correct chart type is recommended) and overall (both XY-axis and chart type are considered). Due to the variations among the above baseline models in calculating metrics, with some evaluations based on individual fields and others on entire tables, for fair comparisons, we report the metrics at the both level. More details are in Appendix C.

Implementations We randomly split the dataset into three parts: 80% for training, 10% for validation, and 10% for testing. For contrast learning, we use CLIP (Radford et al., 2021) visual encoder to extract features of input images. For our chart generation model, we primarily utilize the pre-trained Bloom (Workshop et al., 2022) model with 1.1B parameters as the backbone. We use K-Means (MacQueen et al., 1967) to cluster table features. Additional details are provided in Appendix D.

5.2 Comparison with Baselines

The performance comparison of our HTP and baselines is presented in Table 1, in which the mean and standard deviation of all metrics are obtained through three random runs. We have the following observations: (1) Our HTP consistently achieves the best performance in most of the evaluation metrics, with 25.4% and 32.8% absolute improvements of overall accuracy in field level and table level. This indicates that our model is capable of processing and integrating subtasks within visualization recommendation from a comprehensive perspective. (2) In chart type accuracy, our model achieves an accuracy of 0.780 at the field level and 0.802 at the table level, with a large margin of 13.1% and 17.8% over the best-performing baseline, respectively. This demonstrates the effectiveness of using style query to enhance table perception across various chart types. Moreover, chart type prediction accuracy is generally lower than xy prediction across all models, suggesting it's more challenging and could be a bottleneck in chart recommendation. (3) Table2Charts exhibits greater accuracy on the xy-axis than other methods. This may be due to the pre-defined chart template constraining the selection of x and y-axis. However, in real-world scenarios with more complex charts, such a strongly constrained template might

Model	Field Level			Table Level		
	XY	Chart Type	Overall	XY	Chart Type	Overall
VizML	0.892 ± 0.008	0.553 ± 0.009	<u>0.507 ± 0.014</u>	0.795 ± 0.018	0.393 ± 0.015	0.334 ± 0.017
KG4Vis	0.819 ± 0.038	0.418 ± 0.055	0.319 ± 0.052	0.552 ± 0.030	0.257 ± 0.014	0.111 ± 0.024
MultiVision	-	<u>0.649 ± 0.020</u>	-	-	<u>0.624 ± 0.021</u>	-
Data2Vis	0.646 ± 0.038	0.341 ± 0.010	0.321 ± 0.012	0.508 ± 0.027	0.415 ± 0.007	0.281 ± 0.014
Table2Charts	**0.932 ± 0.015**	0.453 ± 0.016	0.436 ± 0.010	**0.881 ± 0.013**	0.426 ± 0.007	<u>0.397 ± 0.012</u>
DeepEye	0.523 ± 0.009	0.396 ± 0.022	0.237 ± 0.006	0.493 ± 0.006	0.427 ± 0.004	0.249 ± 0.004
HTP	<u>0.928 ± 0.011</u>	**0.780 ± 0.014**	**0.761 ± 0.013**	<u>0.874 ± 0.009</u>	**0.802 ± 0.011**	**0.725 ± 0.008**

Table 1: Performance comparison between HTP and all the baselines. The best results are in **bold** and the second are <u>underlined</u>.

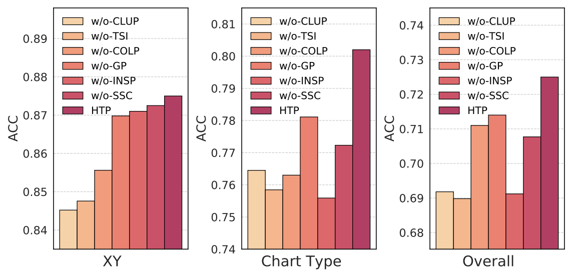

Figure 3: Performance comparison between HTP and its variants on Table Level.

lack flexibility. (4) KG4Vis has high standard deviation across all metrics, and a large drop in overall accuracy from field to table level. This demonstrates that focusing solely on single-column features while neglecting the comprehensive attributes of the entire table leads to incomplete understanding, consequently impairing performance at the table level. In contrast, our HTP , by encompassing information across all table levels, achieves higher accuracy with exhibiting great consistency.

5.3 Ablation Analysis

To verify the effectiveness of each design in our model, we further compare HTP with six variants:

- **w/o-CLUP** removes the cluster-level prompt.
- **w/o-TSI** removes the all table structure information (include the column-level prompt).
- **w/o-COLP** removes the column-level prompt.
- **w/o-GP** removes the general prompt.
- **w/o-INSP** removes the instance-level prompt.
- **w/o-SSC** removes the soft codes, using only hard codes as source style query.

According to the results shown in Figure 3, HTP outperforms all its variants, proving the significance of our special designed prompts. Specifically, w/o-INSP and w/o-TSI underperform other variants, highlighting the need for dynamically generating customized prompts to capture fine-grained information and the internal structure of tables. Meanwhile, we can see w/o-INSP has a significant performance degradation on the chart type prediction, proving that instance-level prompt can effectively obtain perceptual information between table and different types of charts. Besides, we observe a decrease when removing the cluster-level prompt, which demonstrates the importance of sharing information within table patterns (w/o-CLUP). In addition, the performance degrades if the soft codes is not used, suggesting a comprehensive understanding of chart representation through capturing intra-class connections is crucial for enhancing model performance. Further, removing the table-column prompt results in a notable performance decline, which verifies the importance of understanding table structure and enhancing column-level insights.

5.4 Comparison with Adaptation Methods across Model Scales

To evaluate the effectiveness and adaptability of our model, we conduct a comprehensive comparison between HTP and the following methods at different model scales, including Fine-Tuning, Prompt Tuning (Lester et al., 2021), Prefix Tuning (Li and Liang, 2021) and LoRA (Hu et al., 2022). From Table 2, we observe that: (1) Across the majority of metrics and model scales, our model significantly outperforms other partial parameter tuning methods. This indicates that our proposed prompt-based framework can comprehensively unearth the implicit insights of tables and more efficiently harness the potential of LLMs for visualization recommendation. (2) Our HTP outperforms parameter-efficient tuning methods by a large margin in small-scale LLMs. For example, HTP achieves absolute improvements of 8.3% and 12.9% over the best-performing PEFT method in overall metric at field level and table level respectively. (3) We can see that there is still a significant gap between Fine-Tuning and parameter-efficient tuning methods. However, our method achieves comparable

Model	Method	Field Level			Table Level		
		XY	Chart Type	Overall	XY	Chart Type	Overall
Bloom-Small (560M)	Fine-Tuning	0.953	0.745	0.734	0.918	0.701	0.649
	Prompt Tuning	0.899	0.550	0.534	0.843	0.549	0.492
	Prefix Tuning	0.924	0.592	0.584	0.869	0.584	0.550
	LoRA	**0.927**	0.624	0.613	**0.887**	0.577	0.536
	HTP	0.902	**0.714**	**0.696**	0.845	**0.746**	**0.679**
Bloom-Medium (1.1B)	Fine-Tuning	0.940	0.782	0.772	0.896	0.803	0.755
	Prompt Tuning	0.876	0.682	0.656	0.834	0.615	0.543
	Prefix Tuning	0.912	0.746	0.726	0.858	0.765	0.697
	LoRA	0.915	0.758	0.739	0.864	0.777	0.703
	HTP	**0.928**	**0.780**	**0.761**	**0.874**	**0.802**	**0.725**
Bloom-Large (3B)	Fine-Tuning	0.950	0.832	0.820	0.915	0.834	0.787
	Prompt Tuning	0.817	0.670	0.672	0.845	0.748	0.676
	Prefix Tuning	0.923	0.841	0.818	0.897	0.804	0.763
	LoRA	0.936	0.813	0.798	0.897	0.820	0.764
	HTP	**0.938**	**0.850**	**0.836**	**0.899**	**0.831**	**0.773**

Table 2: Performance comparison between HTP and adaptation methods at different model scales. The best results except Fine-Tuning are in **bold**, and the second are underlined.

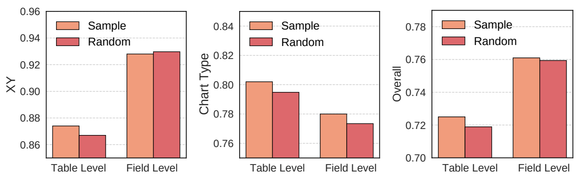

Figure 4: Performance comparison between different prompt initialization methods.

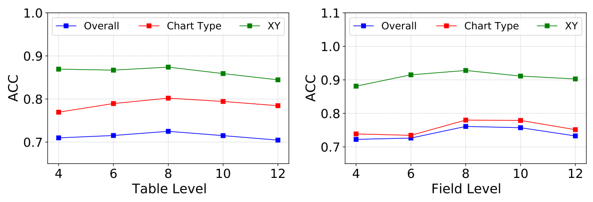

Figure 5: Parameter sensitivity of the number of table pattern clusters K.

performance to Fine-Tuning across models of various scales, and even outperforms Fine-Tuning in most metrics. In the 560M model, both the chart type and overall accuracy at table level exceed Fine-Tuning. Moreover, in the 3B model, these two metrics also surpass Fine-Tuning performance at the field level. This demonstrates that HTP can better enrich the semantic features of table data and align highly structured table data with the pre-training paradigm of LLMs without loss of information.

5.5 Parameters Analysis

5.5.1 Prompt Initialization

We explore the impact of soft prompt initialization in our study. Our investigation focuses on two distinct initialization strategies: (1) *Random-based Initialization.* Soft prompts are generated by randomly initializing their embeddings. (2) *Sample Vocabulary-based Initialization.* We sample high-frequency word chunks from the training set dictionary, concatenate them in order of frequency, then trim to match the prompts' length. We use the embeddings of the trimmed version as initial values. As shown in Figure 4, we can see that HTP is robust to the prompt initialization method, achieving comparable results with both initialization choices.

5.5.2 Number of Table Patterns Clusters K

As depicted in Figure 5, more clusters of table pattern, namely increasing K, enables HTP to capture more complex and diverse table pattern information. However, if K gets too large, there may be insufficient corresponding table groups in the table dataset to support cluster representation learning, and some superfluous clusters may introduce noise and undermine performance instead.

6 Conclusion

In this paper, we introduced a novel Hierarchical Table Prompt-based reprogramming Framework, called HTP, to enhance the visualization recommendation process through Large Language Models (LLMs). The HTP harnessed potential of LLMs through foul level of prompts, aiming at extracting semantic information from table through comprehensive perspectives. In this way, HTP effectively unlocked the potential of LLM for transforming table into insightful charts. Extensive experimental results demonstrated the superior performance of HTP framework.

7 Limitations

We briefly mention some limitations of our work. First, we have only used a single self-collected dataset. This is mainly due to the fact that the only open-source dataset Plotly Corpus, collected by (Hu et al., 2019), was not accessible. Moreover, we mainly consider data-encodings attributes including visualization types and the arrangement of x-/y-axes while non-data-encoding attributes such as layouts and colors are not considered, which is because the evaluation of non-data-encoding attributes is relatively subjective, and our dataset predominantly utilizes default settings for these attributes.

References

Miao Chen, Xinjiang Lu, Tong Xu, Yanyan Li, Jingbo Zhou, Dejing Dou, and Hui Xiong. 2022. Towards table-to-text generation with pretrained language model: A table structure understanding and text deliberating approach. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8199–8210.

Victor Dibia and Çağatay Demiralp. 2019. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications*, 39(5):33–46.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*.

Kevin Zeng Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César A. Hidalgo. 2019. Vizml: A machine learning approach to visualization recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 128.

Baoshuo Kan, Teng Wang, Wenpeng Lu, Xiantong Zhen, Weili Guan, and Feng Zheng. 2023. Knowledge-aware prompt tuning for generalizable vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15670–15680.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Nataliya Kosmyna, Jussi T Lindgren, and Anatole Lécuyer. 2018. Attending to visual stimuli versus performing visual imagery as a control strategy for eeg-based brain-computer interfaces. *Scientific reports*, 8(1):13222.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Hao-Tian Li, Yong Wang, Songheng Zhang, Yangqiu Song, and Huamin Qu. 2021. Kg4vis: A knowledge graph-based approach for visualization recommendation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1).

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597.

Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. 2018. Deepeye: Towards automatic data visualization. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 101–112.

Jock Mackinlay. 1986. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141.

Jock Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show me: Automatic presentation for visual analysis. *IEEE transactions on visualization and computer graphics*, 13(6):1137–1144.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297.

Daniel Perry, Bill Howe, Alicia Key, and C. Aragon. 2013. Vizdeck: Streamlining exploratory visual analytics of scientific data.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503.

Jorge Poco and Jeffrey Heer. 2017. Reverse-engineering visualizations: Recovering visual encodings from chart images. *Computer Graphics Forum*, page 353–363.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the*

38th International Conference on Machine Learning, pages 8748–8763.

Steven F Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. 1994. Interactive graphic design using automatic presentation knowledge. In *Conference on Human Factors in Computing Systems*, pages 112–117.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, page 341–350.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. Towards table-to-text generation with numerical reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1451–1465.

Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, page 649–658.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Aoyu Wu, Yun Wang, Mengyu Zhou, Xinyi He, Haidong Zhang, Huamin Qu, and Dongmei Zhang. 2021. Multivision: Designing analytical dashboards with deep learning based recommendation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):162–172.

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629.

Mengyu Zhou, Qingtao Li, Yuejiang Li, Shi Han, and Dongmei Zhang. 2020. Table2charts: Learning shared representations for recommending charts on multi-dimensional data. *arXiv preprint arXiv:2008.11015*.

A Data Cleaning

We developed our dataset using a data cleansing pipeline, outlined as follows:

(1) Format Verification. Since the Plotly community stores table and chart data in json format, we will initially filter out data that cannot be parsed in json format.

(2) Completeness Assessment. Data samples lacking either non-empty source table data or chart data are discarded. Additionally, if data columns corresponding to the x and y axes in the chart data are missing in the table data, if table data contains empty columns, or if chart data lacks specified chart type or axes, the sample is excluded.

(3) Data Deduplication. In the Plotly community, each visualization pair is uniquely identified by a "fid". When encountering multiple data entries with the same "fid", only the first one is retained. Since many tables are slight modifications of each other, we calculate their basic features, integrate these values into a single feature identifier, and retain only the first entry for each set of samples sharing this feature identifier.

B Baselines

To evaluate the proposed framework, we adopt six baselines for comparison. Here are the descriptions of these baselines:

- VizML (Hu et al., 2019), which formulates visualization recommendation task as a series of classification challenges, deploying distinct Neural Network-based models for each classification task. We train VizML in our dataset on its Mark Type task (chart type predixtion task and Is on X-axis or Y-axis task). Due to VizML training by columns on its Mark Type task and dividing the dataset at the column level in its source code, it is possible that different columns from the same table could be allocated to different sets(e.g. the first column in the table is classified to the training set while the second column is classified to the test set), which potentially introducing bias. In our training process, we divide the dataset by tables, meaning all columns from the same table are assigned to the same set.
- KG4Vis (Li et al., 2021), which leverages knowledge graphs to recommend from dataset-visualization pairs.
- MultiVision (Wu et al., 2021), which design two scoring network for recommending single and multiple-view visualizations. Because we do not recommend multiple-view visualizations, only the first network is used to train on our dataset. Specially, due to the incomplete disclosure of all training parameters, we set the batch size to 4096 and the learning rate to 3e-3, conducting training

over 30 epochs. Because our dataset contains six types of charts, so we set the initialization parameter num_class of the ChartTypeLSTM model to 6. The remaining model hyperparameters are consistent with those in the source code.

- Data2Vis (Dibia and Demiralp, 2019), which leverages an LSTM-based neural translation model to generate json encoded visualizations in an autoregressive way. Following Data2Vis, for each table-chart pair, three training samples are generated by sampling three rows from table (three different data rows with the same encoded chart), resulting in a total of $279,609$ pairs which are used for training. The vocabulary sizes of source and target are 95 and 38.
- Table2Charts (Zhou et al., 2020), which use deep Q-Network to fill the predefined chart templates by estimating the next token or action. Specificallybecause the chart templates defined in table2charts—scatter, line, bar, and pie match the types of charts in our dataset, we retain only these four types of data for training.
- DeepEye (Luo et al., 2018), which provides two public models (ML and rule-based) without training scripts. Thus, we evaluate its models on our test set and report the best results between two models.

For all baselines, all hyperparameter settings were based on the values reported in the original paper for optimal results, unless otherwise specified.

C Evaluation Details

Let N denote the total number of tables, C_i denote the number of columns in table i, $correct_{xy}^{ij}$ and $correct_{type}^{ij}$ as binary indicators for column j in table i, where $correct_{xy}^{ij}$=1 if the xy axis allocation for column j is correct and 0 otherwise. Similarly, $correct_{xy}^{ij}$=1 if the chart type for column j is correctly predicted, and 0 otherwise.

At the table level, when calculating metrics, the entire table is treated as a single unit; a prediction is only considered correct if the X-Y axis allocation and chart type for all columns within the table are accurately predicted, as follows:

$$
\begin{aligned}
Acc_{xy} &= \frac{\sum_{i=1}^{N}(\prod_{j=1}^{C_i} correct_{xy}^{ij})}{N}, \\
Acc_{chart\ type} &= \frac{\sum_{i=1}^{N}(\prod_{j=1}^{C_i} correct_{type}^{ij})}{N}, \\
Acc_{overall} &= \frac{\sum_{i=1}^{N}(\prod_{j=1}^{C_i}(correct_{xy}^{ij}\ \&\ correct_{type}^{ij}))}{N}.
\end{aligned}
\tag{7}
$$

While at the field level, we calculate metrics on a per-column basis:

$$
\begin{aligned}
M &= \sum_{i=1}^{N} C_i, Acc_{xy} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{C_i} correct_{xy}^{ij}}{M}, \\
Acc_{chart\ type} &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{C_i} correct_{type}^{ij}}{M}, \\
Acc_{overall} &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{C_i}(correct_{xy}^{ij}\ \&\ correct_{type}^{ij})}{M}.
\end{aligned}
\tag{8}
$$

D Implementations

To prevent bias towards imbalanced data, we randomly split the dataset by chart type, allocating 80% for training, 10% for validation, and 10% for testing for each chart category. We run all experiments on NVIDIA RTX A100 GPUs, and use the Adam (Kingma and Ba, 2015) optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to optimize models. For contrast learning , we adopt CLIP's (Radford et al., 2021) visual encoder to extract the global features of input images. The temperature parameter, memory bank size, batch size and learning rate are set to 0.07, 2048, 1024, and 9e-4, respectively. We employ K-Means (MacQueen et al., 1967) to cluster table features, to minimize the impact of random initialization of initial cluster centers as much as possible, we execute the algorithm five times and select the iteration with the lowest Sum of Squared Errors (SSE) as the final result. For our chart generation model, we primarily utilize the pre-trained Bloom (Workshop et al., 2022) model with 1.1 billion parameters as our backbone. For training our chart generation model, we set the learning rate, batch size to 1e-5 and 4. The length of general-level, instance-level, column-level, cluster-level prompts are set to 20, 100, 1, and 15. In addition to parametric experiments, we fixed the number of clusters for cluster prompt at 8 and the prompt initialization method to Sample Vocabulary-based Initialization. In the generation stage, we adopt *top-p* sampling as the default decoding method with a temperature of 0.1 and a *top-p* = 0.75.

E Table Features for Clustring

We use table features that introduced in VizML (Hu et al., 2019) for clustering. Following VizML, we first extract 81 single-column features, comprising both 50 continuous features and 31 categorical features. These features are categorized into four main groups: the data type of the column (Types), statistical characteristics of the column's values such

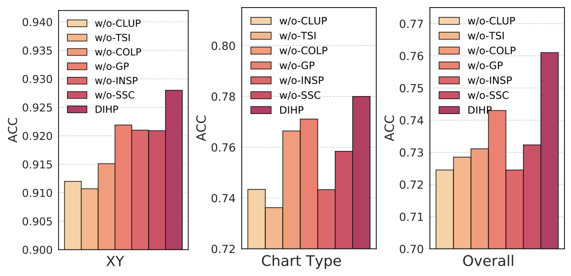

Figure 6: Performance comparison between HTP and its variants on Field Level.

as distribution and outliers (Values), the column's name (Names), and the column's row count (Dimensions). To describe the relationship between pairs of columns, we employ 30 pairwise-column features. In the final step, we utilize 16 aggregation functions to combine both pairwise-column and single-column features, yielding 841 dataset-level features for clustering.

F More Results about Ablation Analysis

Due to space constraints and the fact that trends at both the table level and field level are largely similar, we have chosen to present only the table level results in the main text. For completeness, we also include the field level results in Figure 6. The analysis of ablation studies at the field level follows a similar pattern to that at the table level.

G Image Dataset

We collect 125,121 chart images from the public web. These include 28100 bar images, 9838 box images, 7319 histogram images, 38725 line images, 37215 scatter images and 3924 pie images.