AdaVis: Adaptive and Explainable Visualization Recommendation for Tabular Data

Songheng Zhang, Yong Wang, Haotian Li, and Huamin Qu, *Member, IEEE*

Abstract—Automated visualization recommendation facilitates the rapid creation of effective visualizations, which is especially beneficial for users with limited time and limited knowledge of data visualization. There is an increasing trend in leveraging machine learning (ML) techniques to achieve an end-to-end visualization recommendation. However, existing ML-based approaches implicitly assume that there is only one appropriate visualization for a specific dataset, which is often not true for real applications. Also, they often work like a black box, and are difficult for users to understand the reasons for recommending specific visualizations. To fill the research gap, we propose *AdaVis*, an adaptive and explainable approach to recommend one or multiple appropriate visualizations for a tabular dataset. It leverages a box embedding-based knowledge graph to well model the possible one-to-many mapping relations among different entities (i.e., data features, dataset columns, datasets, and visualization choices). The embeddings of the entities and relations can be learned from dataset-visualization pairs. Also, *AdaVis* incorporates the attention mechanism into the inference framework. Attention can indicate the relative importance of data features for a dataset and provide fine-grained explainability. Our extensive evaluations through quantitative metric evaluations, case studies, and user interviews demonstrate the effectiveness of *AdaVis*.

Index Terms—Visualization Recommendation, Logical Reasoning, Data Visualization, Knowledge Graph.

✦

1 INTRODUCTION

D ATA visualization has become increasingly popular in data analytics and insight communication. It is common to create visualizations for tabular datasets in various domains, including investment, sales, engineering, education, and scientific research [1], [2]. However, creating compelling visualizations requires expertise in data visualization and relies on manual specifications through either programming or mouse interactions (e.g., clicking and dragging, and dropping). The visualization tools can generally be categorized into two types: visualization packages (e.g., ggplot2 [3], Vega [4], D3 [5], and Prefuse [6]) and visualization software (e.g., Tableau [1] and Microsoft Power BI [2]). The former needs users to do programming with different languages (e.g., Python, R, Java, and JavaScript), and the latter often asks users to manually drag and drop and specify the mapping between data and visual encodings. As a result, it is often complicated and time-consuming for common users without a background in data visualization to generate effective visualizations.

Automatic visualization recommendation techniques have been proposed to make the visualization creation process more accessible and more efficient [7], [8], [9], [10], [11]. Among them, the machine learning (ML) based visualization recommendation approaches have become popular in the past few years. They are often data-driven and implicitly model the mapping between datasets and appropriate visualizations. Compared with other rule-based visualization recommendation techniques (e.g., APT [12], Show Me [13] and SeeDB [14]), the ML-based approaches have the advantage of being an end-to-end visualization recommendation without

specifying heuristics, and there have been an increasing number of research studies to achieve better visualization recommendations by leveraging machine learning techniques [15], [16], [17].
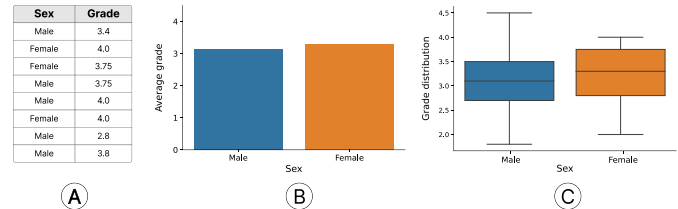

Figure 1. A dataset regarding students' grades in a class. The dataset can be visualized in a bar chart or a box plot.

However, existing ML-based visualization recommendation approaches suffer from two major issues: ***adaptability*** and ***explainability***. First, the ML-based approaches implicitly assume a one-to-one correspondence between datasets and visualizations by training on dataset-visualization pairs. However, it does not always hold in real applications. For example, as shown in Figure 1, the dataset that contains students' grades over a course can be visualized as either a bar chart showing the average grades of female and male students or a boxplot displaying the grades distribution of female and male students. Both visualizations are suitable in terms of visual encodings and the optimal choice can be either of them, depending on users' preferred level of details of GPA distribution.

According to our survey, no existing ML-based approaches can adaptively recommend multiple appropriate visualizations for a dataset. Existing ML-based approaches often recommend the only one visualization choice [1], [7], [9], [18], not adaptive to different datasets. Second, most ML-based approaches (e.g., Data2Vis [7] and VizML [9]) are built upon deep neural networks. For instance, a three-layer fully connected neural network is employed in VizML [9] to predict the axis encodings and visualization types for a dataset. These approaches work as a black box and can

- *S. Zhang, Y. Wang are with the School of Computing and Information System, Singapore Management University, Singapore. E-mail: {shzhang.2021, yongwang}@smu.edu.sg.*
- *H. Li and H. Qu are with the Hong Kong University of Science and Technology E-mail: haotian.li@connect.ust.hk, huamin@cse.ust.hk.*

1. https://www.tableau.com/
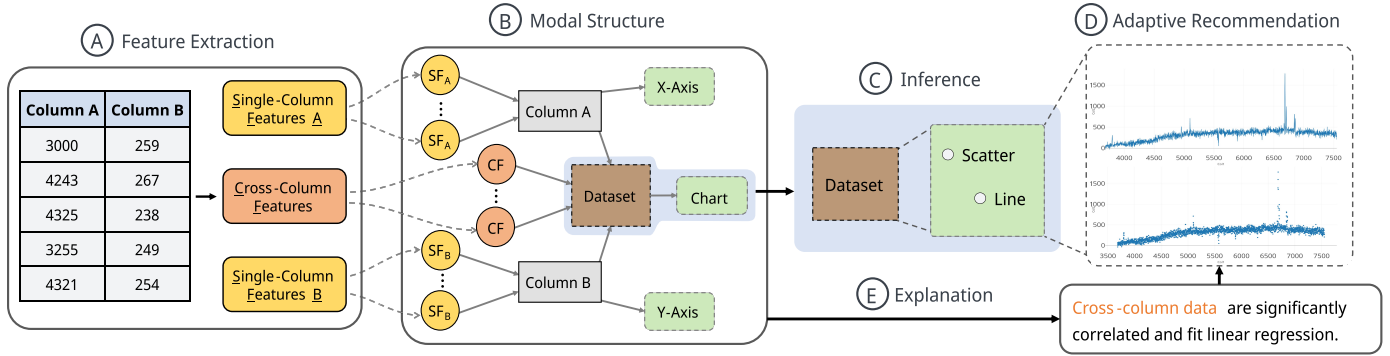2. https://powerbi.microsoft.com/en-us/desktop/

Figure 2. The workflow of *AdaVis* recommendation consists of Feature Extraction, Model Structure, Inference, Explanation Generation and Adaptive Recommendation. (A) delineates that *AdaVis* extracts cross-column and single-column features from a dataset; (B) displays the structure of the *AdaVis*: it uses extracted features to infer appropriate visualization choices; (C) illustrates the inference details: the visualization types within a box are identified as visualization type recommendations for the dataset; (D) displays the multiple appropriate visualization types for the dataset; (E) illustrates that *AdaVis* provides an explanation for the recommendation results.

undermine users' trust in the visualization recommendation results, especially for general users without a background in visualization and deep neural networks. A recent study, KG4Vis [18], presents a knowledge-graph-based approach to recommend visualization in an explainable manner for tabular datasets. But their explanations are built on single-column features and can only indicate which single-column features account more for a visualization type, which is called *global interpretation* [19]. The global interpretation captures the overall relationship the model learned from all datasets, identifying which feature values are strongly associated with specific visualization types. However, KG4Vis neglects cross-column features and fails to provide a fine-grained explanation for each unique dataset. The *local interpretation* [19] that focuses on explaining a particular visualization recommendation for an individual dataset is still missing

In this paper, we propose *AdaVis*, an **Ad**aptive and Explainable **Vis**ualization Recommendation approach for tabular data through logical reasoning over knowledge graphs. Inspired by KG4Vis [18], our approach also leverages a knowledge graph to model the relations between different entities involved in visualization recommendation, e.g., *data features*, *dataset columns*, *datasets* and *visualization design choices*. The relations in the knowledge graph define the correspondence between two different types of entities. For example, "(a dataset) is visualized by (a visualization choice)". Such relations intrinsically specify the inference rules in visualization designs. However, instead of employing the widely-used vector embeddings [18], [20] to indicate the inference results, we adopt box embeddings [21] that essentially allow the visualization recommendation results to cover multiple appropriate visualization choices for a given dataset. The incorporation of box embeddings leads to better ***adaptability*** for visualization recommendation, enabling *AdaVis* to adaptively recommend an appropriate number of visualization choices based on the characteristics of a dataset. Also, we have incorporated an attention mechanism into *AdaVis*, which assesses the importance of different features for visualization recommendations [22]. This mechanism works over the knowledge graph, ensuring that our recommendations are informed by relevant data features. Moreover, *AdaVis* offers fine-grained explanations for the visualization recommendations for a specific dataset (*local interpretation*) by tracing the importance of data features along inference paths, thereby improving the interpretability of our recommendations. The explanations are natural language (NL) sentences automatically generated from rule-based templates.

We extensively evaluated the effectiveness and usability of *AdaVis* by using the dataset-visualization pairs collected by Hu et al. [9]. We first quantitatively compared *AdaVis* with other state-of-the-art baseline approaches in terms of visualization recommendation accuracy. Then, we showed a gallery of visualization recommendation results and the corresponding natural language explanations to demonstrate the adaptability and explainability of *AdaVis*. Further, we conducted user interviews to invite both data visualization experts and common users to verify whether the recommended visualizations are meaningful and align well with their domain knowledge of visualization design requirements and whether explanations regarding these recommendations are correct.

The paper's main contributions can be summarized as follows:

- We propose *AdaVis*, an adaptive and explainable visualization recommendation approach for tabular data via knowledge graphs. It adaptively recommends multiple appropriate visualizations for a specific dataset, better modeling the real visualization design process. Also, it can provide fine-grained explanations for different datasets.
- We extensively assess *AdaVis* through quantitative metric comparisons with other baseline approaches, qualitative case studies, and user interviews. The results demonstrate the effectiveness and usability of *AdaVis* in providing adaptive and explainable visualization recommendations.

2 BACKGROUND: BOX EMBEDDING IN KNOWLEDGE GRAPHS

Knowledge Graphs and Relations. A knowledge graph models human knowledge as a directed graph with entities and relations. Each entity is a graph node, and each relation is a graph edge. The relationship between any two entities is delineated by a triple (h, r, t), where h represents a head entity, t represents a tail entity, and r represents a relation. Most relationships in a knowledge graph are 1-to-1 mappings, with the relation r between the head entity h and the tail entity t being unique. For example, *"US has a citizen named Bob"* is an example of the 1-to-1 relation, as shown in Figure 3(c), and the corresponding triple is *(US, Has Citizen(s), Bob)*. Besides, there are 1-to-N relations in a knowledge graph. For instance, *"Amazon has employees Alice and Bob"* represents a 1-to-N relation. Figure 3(a, c) delineates the 1-to-N relation, where there are more than one tail entities (i.e., *Alice* and *Bob*)
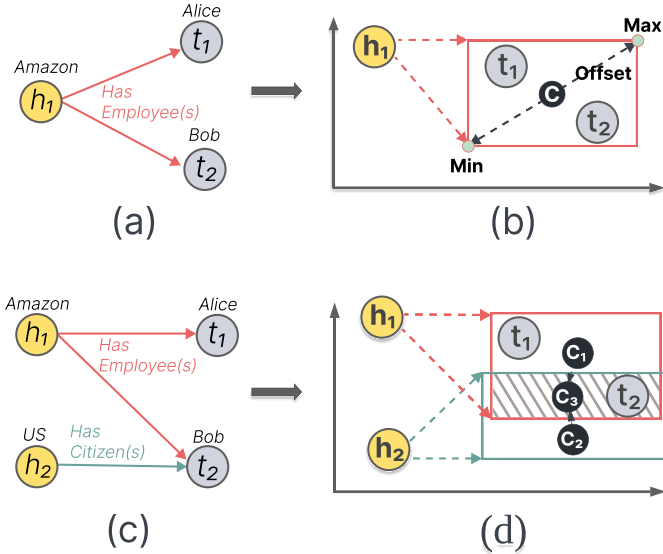
Figure 3. Illustration of the box embedding two operations: projection and intersection in a knowledge graph (a, c) and vector space (b, d), respectively. h_1 and h_2 are the head entities representing a company Amazon and a country US, respectively. Similarly, t_1 and t_2 are tail entities representing two individuals in a KG. The red and green arrows are different relations between head entities and tail entities. Additionally, **h** and **t** are entities' vector forms in the vector space. Besides, **C** denotes the center of the box. **Offset** represents the box range, thereby determining the box's size. **Max** and **Min** are the endpoints of the box's diagonal. (a) displays a 1-to-N relation triple in which *Amazon* (h_1) has *employees* named *Alice* (t_1) and *Bob* (t_2). (b) demonstrates the 1-to-N triple computation in the 2-D vector space by the projection: the relation *(Has Employee(s))* projects the head entity into a box that contains the tail entities. (c) indicates the intersection of two 1-to-N triples where *Bob* (t_2) is an *employee* of *Amazon* (h_1) and a *citizen* of *US* (h_2). (d) shows the box intersection in the 2-D vector space: two boxes' intersection gives rise to a smaller box that contains a tail entity (**t₂**) relevant to both two triples. Otherwise, the irrelevant tail entity (**t₁**) is outside the smaller box. The center **C₃** of the smaller box is based upon the two projected box centers **C₁** and **C₂**.

for the same head entity and relation. The triple is *(Amazon, Has Employee(s), {Alice, Bob})*.

Box Embedding. To facilitate computational manipulations on knowledge graphs, it's often necessary to apply Knowledge Graph Embedding (KGE) to represent entities and relations in KGs as continuous embedding vectors [23]. For the 1-to-1 relations in knowledge graphs, there have been many KGE methods to model them, e.g., TransE [24] and PTransE [25]. However, they cannot model the 1-to-N relations that entail a set of tail entities as shown in Figure 3(a). Also, these methods cannot be used to define the intersection of multiple 1-to-N triples. The intersection of 1-to-N triples will obtain a set of common tail entities. These common tail entities are relevant to all these 1-to-N triples. For example, Figure 3(c) shows the intersection of two triples, where *"Bob is an employee of Amazon and also a US citizen"*. To handle these challenges in a scalable manner, box embedding is introduced recently [26]. Rather than representing a point in vector space, it represents an area and can handle 1-to-N relations and the intersection of 1-to-N relations using two operations: *projection* and *intersection*. The projection operation of box embedding maps an entity embedding (i.e., a point) to a box area (i.e., axis-aligned hyper-rectangles) in the vector space (Figure 3(b)). Tail entities should be enclosed within the projected box and satisfy the following condition:

$$Box \equiv \{\mathbf{v} \in \mathbb{R}^d : Cen(Box) - Off(Box) \preceq \mathbf{v} \preceq Cen(Box) + Off(Box)\}, \quad (1)$$

where $\preceq$ denotes element-wise inequality, $Cen(Box) \in \mathbb{R}^d$ denotes the center point of the box and $Off(Box) \in \mathbb{R}^d \geq 0$, which stands for the positive offset of the box. The offset indicates the size of the projected box, as shown in Figure 3(b).

The intersection operation of box embeddings models the intersection of multiple 1-to-N relations by intersecting several projected boxes. For instance, Figure 3(c) depicts the intersection of two triples, where h_1 and h_2 have different relations to t_2. The intersection of two box embeddings projected from **h₁** and **h₂**, shown by the small shadowed box in Figure 3(d), identifies the t_2 to which both h_1 and h_2 have relations. **t₂** is within the intersected box, indicating that t_2 is the tail entities of both two triples.

For visualization recommendations, it is common for a dataset to be represented as multiple visualizations types, which is a 1-to-N relation. To model these relations accurately, we utilize box embedding in our approach for adaptive visualization recommendations.

3 RELATED WORK

The related work of this paper can be categorized into three groups: visualization recommendation, knowledge graph embedding, knowledge graph-based explainable recommendation.

3.1 Visualization Recommendation

Visualization recommendation aims to suggest or generate appropriate visualizations for a given dataset automatically and generally includes two types of methods [15]: Rule-based methods and machine learning (ML)-based approaches.

Rule-based methods leverage visualization rules specified by visualization experts to recommend appropriate visualizations [13], [14], [27]. For example, Mackinlay *et al.* proposed *Show Me*, which can automatically suggest visualizations using predefined visualization guidelines [13]. Using a predefined set of rules, voyager [27] and voyager2 [28] enumerates all potential data columns in a dataset to get potential visualizations and further ranks all these visualizations to recommend appropriate choices. Additionally, Foresight [29] detects pre-defined statistical features from the dataset and then made recommendations according to these features. and present them visually through appropriate chart types. Though rule-based methods have been extensively studied, developing a comprehensive list of rules for visualization recommendations is challenging, and the maintenance of such empirical rules is often labour-intensive [18].

ML-based methods learn the mappings between input datasets and visualizations [15] from training examples. For instance, Vizdeck [30] trains a linear model for this mapping. DeepEye [31] uses a *learning-to-rank* [32] model to rank visualization recommendations, then recommends the top scoring one. Draco [33] employs the statistical model RankSVM to rank possible visualizations. More recently, deep neural networks have also been widely used for visualization recommendations, such as VizML [9], Data2Vis [7] and Table2Charts [1]. While these ML-based methods can reduce the manual efforts of compiling rules for visualization recommendations, they often operate like a black box, making it difficult for general users to interpret them [15]. Li *et al. [18] proposed a knowledge graph-based recommendation approach. Their approach recommends suitable visualization choices in a*

data-driven and explainable manner, making it the most relevant study to our work. However, this approach fails to consider relationships between data columns in the dataset, which is crucial for determining visualization choices.

Unlike the above studies, *AdaVis* takes into account the cross-column relationships of the input dataset and can provide adaptive visualization recommendations and explanations.

3.2 Knowledge Graph Embedding

Knowledge Graph (KG) models the relations between different entities [34], and knowledge graph embedding (KGE) maps entities and relations into embedding vectors while preserving their semantic meanings, which mainly includes semantic matching models and translational distance models [23]. Semantic matching models evaluate the plausibility of a triple by matching the entities and relations with latent semantics in the vector space. For example, RESCAL [35] assigns a vector embedding to each entity in the knowledge graph, and each relation is interpreted as a matrix that models the semantic interaction between two entities. Dismult [36] restricts the relation matrix of RESCAL to multiple diagonal matrices, thereby simplifying the calculation of RESCAL.

Translational distance models use translation operations to represent the relations between any two entities, where the distance between the entity embedding after a translation and the other entity embedding indicates the plausibility of a triple. TransE [24] is one of the most representative translational distance methods. When using TransE, combining the embedding vector of a head entity with that of a relation creates a new embedding in the vector space that approximates the tail entity. The limitation of TransE is that it implicitly assumes that there are only one-to-one relationships between entities and cannot deal with 1-to-N relationships [37]. Therefore, other methods have been proposed to improve the modeling of 1-to-N relationships in knowledge graphs [21], [37], [38], [39]. For example, query2box [21] introduces box embeddings whereby a head entity embedding can be translated into a box by a relation embedding. rather than a point in the vector space. When the embedding vector of a tail entity lies inside the projected box embedding of a head entity, the corresponding triple is considered valid, making it able to represent 1-to-N relations.

Our approach is inspired by query2box [21] and incorporates box embedding in our knowledge graph to model the 1-to-N and intersection of 1-to-N relations in visualization recommendation. Also, we augment the original loss function of query2box to enhance the adaptability of recommended visualizations.

3.3 Knowledge Graph Based Explainable Recommendation

Knowledge graphs have been integrated into recommendation systems to enhance their interpretability [34]. According to the survey by Li *et al.* [40], the knowledge graph-based explainable recommendation methods can be grouped into two categories: Internal route-based methods and external route-based methods. For the internal route-based methods, the recommendation algorithms are designed by explicitly considering the knowledge graphs, including their entities, relations, paths, and rules, to improve the recommendation performance and provide explanations. For instance, Wang *et al.* [41] defined the relations between entities as sequential paths and further leveraged a Recurrent Neural Network (RNN) to model the sequential dependencies of entities within a

knowledge graph. Also, Ma *et al.* [42] directly derived recommendation rules from the knowledge graph and recommended items based on the extracted rules.

In contrast, external route-based recommendation methods are not built upon knowledge graphs. Instead, they only use external knowledge graphs to generate explanations for the recommendation results. For example, the medical knowledge graph has been used to discover possible explanations for previous medical treatments [43]. Also, Sarker *et al.* [44] utilized an external knowledge graph to elucidate the behaviors of neural network classifications.

Our approach falls under internal routes-based recommendation methods, and integrates a knowledge graph into our recommendation framework. Tracing back paths in the knowledge graph can provide meaningful explanations for the recommended visualizations.
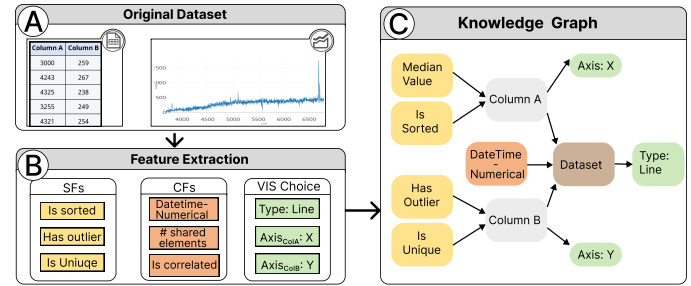

Figure 4. An example of transforming a dataset into a knowledge graph. (A) The dataset contains a pair of tabular data and corresponding visualization. (B) With feature extraction, the individual data columns' characteristics, namely single-column features (SFs) and the interrelationships of two data columns, namely cross-column features (CFs) will be obtained, as well as the mapping between the dataset and visualization choices (VIS Choice). Visualization choices include the visualization type (e.g., line chart) and the axis (e.g., x-axis). (C) The single-column features , cross-column features , data columns , datasets , and visualization choices are represented as entities in the knowledge graph. Only part of the features and entities are shown.

4 OUR METHOD

We propose *AdaVis*, an adaptive and explainable knowledge-graph-based approach to recommend visualizations for tabular datasets. Given that the choice of standard visualization types (i.e., line chart, bar chart, box plot, and scatter plot) often depends on the two data columns displayed on the chart axes, we formulate the visualization recommendation problem for two-dimensional datasets as logical reasoning over the KG to infer visualization types for two-column datasets [21], [45]. The source code of our approach will be publicly released to benefit the broader research community once the paper is published.

4.1 Overview

When determining appropriate visualizations for a dataset, users often need to consider the characteristics of two data columns of interest and their interrelationships. The design of *AdaVis* is inspired by the logical reasoning process of humans when they select the right visualizations. Such a reasoning process is modeled by a knowledge graph consisting of entities (i.e., single-column features, cross-column features, data columns, datasets, and visualization choices). In this paper, we refer to an individual column of a dataset as a data column. A single-column feature is a quantified characteristic of a data column. Similarly, a cross-column feature is a quantified interrelationship between data columns.
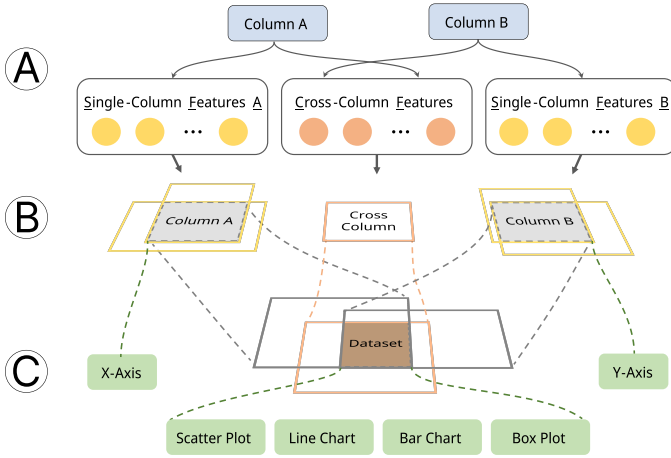
Figure 5. This figure illustrates the workflow of *AdaVis* infers appropriate visualization types. Ⓐ illustrates that we extract single-column and cross-column features from a dataset. Each node represents a feature entity. Ⓑ describes the procedure for generating data column and dataset box embeddings in inference. A hollow yellow/ orange rectangle represents a box embedding from a single-column/ cross-column feature. A gray rectangle denotes a data column obtained from an intersection of box embeddings of single-column features. The brown rectangle represents the dataset generated from an intersection of box embeddings that represent data columns and cross-column features. Ⓒ indicates that box embeddings of data columns and dataset can infer correct visualization choices (i.e., axis and visualization types).

AdaVis comprises feature extraction, knowledge graph construction, box embedding learning, model inference, and explanation generation for the visualization recommendation, which collectively facilitate visualization recommendation. Given a corpus of dataset-visualization pairs, *AdaVis* can learn the implicit mapping between datasets and visualization choices (i.e., visualization types and an axis), which will be further used for visualization recommendations for new datasets. Specifically, as shown in Figure 4 Ⓐ and Ⓑ, *AdaVis* extracts related features for columns in a dataset. Data column features, data columns, datasets, and visualization choices will be used to construct knowledge graphs (Figure 4 Ⓒ). Further, we utilize the box embedding technique [21] to learn the embeddings of entities in the knowledge graph, which can well model their relations in visualization recommendations. We will use the learned embeddings of the knowledge graph's entities and relations to infer suitable visualization choices (i.e., visualization types or an axis) for an unseen dataset. Moreover, a template-based explanation module, built upon the knowledge graph, is integrated into *AdaVis* to generate natural language explanations for the visualization recommendation.

4.2 Feature Extraction

Visualization choices depend on the characteristics of the input dataset. To extract quantified characteristics (features) of datasets, we first surveyed prior studies of visualization recommendation [9], [30], [31] and visualization insight discovery [46], [47]. Based on the survey results, we categorized the dataset features into two types: *single-column features* and *cross-column features*, as shown in Figure 4 Ⓐ and Ⓑ.

The *single-column features* describe the properties of individual data columns, such as the column length, the mean and variance of a column's values. We extracted 80 distinct single-column features that can collectively model the properties of individual data columns from various perspectives. Besides the 80 single-column features, we also extracted 40 well-designed cross-column

features in *AdaVis* by referring to prior studies [9], [31], [46]. The *cross-column features* capture the interrelationships between two columns, e.g., two columns' data types are categorical and numerical. Such cross-column features are also crucial for deciding the visualization choices. For example, compared with a line chart, it is more appropriate to use a scatter plot to visualize a two-column dataset with two columns exhibiting significant correlation [48]. Furthermore, as shown in Figure 4 Ⓑ, we obtain the visualization choices (i.e., visualization types and axes) of datasets. A complete list of all the features used in *AdaVis* can be found in the appendix.

4.3 Knowledge Graph Construction

A knowledge graph allows us to model the mapping between datasets and different visualization choices. With a well-designed knowledge graph, we can further recommend appropriate visualizations.

Definition of Entities. As shown in Figure 4 Ⓒ, we define five classes of entities that are encoded with different colors: single-column features ($\mathbb{E}_{SF}$, yellow nodes), data columns ($\mathbb{E}_{COL}$, gray nodes), datasets ($\mathbb{E}_{DS}$, brown nodes), cross-column features ($\mathbb{E}_{CF}$, orange nodes) and visualization choices ($\mathbb{E}_{VIS}$, green nodes).

As shown in Table 1, $\mathbb{E}_{SF}$ represents features extracted from individual data columns; $\mathbb{E}_{CF}$ are cross-column features; $\mathbb{E}_{COL}$ and $\mathbb{E}_{DS}$ refer to data columns and datasets, respectively; $\mathbb{E}_{VIS}$ refers to the choices available for visualizations, and consists of four popular charts (i.e., bar chart, line chart, scatter plot, and a box plot) [49]. as well as the two commonly used axes (i.e., the x-axis and y-axis).

Since single-column and cross-column features can be continuous values, we discretize them into different intervals to represent them as entities in a knowledge graph. Specifically, we utilize the widely-used MDLP approach [50] to transform continuous features into categorical features.

Definition of Relations. As illustrated in Table 1, there are five relations classes in our knowledge graph. (1) $\mathbb{R}_{SF \to COL}$ denotes a class of relations that associate single-column features with single data columns, and this class indicates that these features are present in a single data column. (2) $\mathbb{R}_{COL \to DS}$ represents a class of relations that link a single data column to a dataset. It shows that datasets contain the data column. (3) Similar to $\mathbb{R}_{SF \to COL}$, $\mathbb{R}_{CF \to DS}$ is a class of relations that indicate cross-column features exist in datasets. For example, $\mathbb{R}_{CF \to DS}$ means that *"(one cross-column feature) exists in columns of (a dataset)"*. (4) $\mathbb{R}_{COL \to VIS}$ shows that single data columns are encoded with a specific axis. For example, $\mathbb{R}_{COL \to VIS}$ means that *"(one data column) is encoded as (x-axis)"*. (5) Similarly, $\mathbb{R}_{DS \to VIS}$ means a dataset is encoded as a visualization type.

Definition of Triples. After defining entities and relations, we generate triples based on existing dataset-visualization pairs. These triples are instances of the defined knowledge graph relations. These triples can be categorized into two types, 1-to-N and intersection of 1-to-Ns, as illustrated in Table 1. As shown in Figure 4 Ⓒ, a 1-to-N triple is constructed by a relation (an arrow) and two types of entities (two nodes with different colors). It is a 1-to-N triple (N$\geq$ 1) because one head entity may correspond to multiple tail entities by a relation. For example, in Figure 4 Ⓒ, "(Has Outlier $\to$ Column B)" denotes a single-column feature (i.e., Has Outlier) could exist in many data columns (i.e., Column B), so, in a triple, the single-column feature (i.e., head) may correspond to many data columns (i.e., tails) by a relation (i.e., $\mathbb{R}_{SF \to COL}$). There are five types of 1-to-N triples since the knowledge graph contains five

Table 1
The table shows two categories of triples in the knowledge graph: 1-to-N and the intersection of 1-to-N. 1-to-N triples can be classified as five relations in the knowledge graph. The intersection of 1-to-N triples intrinsically combines multiple 1-to-N relations.

Type	Relations	Meanings	Examples
1-to-N	$\mathbb{R}_{SF \to COL}$	Data columns with the specific single-column feature are	The column (COL) length is 50 (SF)
	$\mathbb{R}_{CF \to DS}$	Datasets with the specific cross-column feature are	Percentage of unique values (CF) shared in a Dataset 's two columns (DS)
	$\mathbb{R}_{COL \to DS}$	Data columns in the specific dataset are	A column representing grades (COL) in a dataset (DS) about students' grades (Figure 1(A)),
	$\mathbb{R}_{COL \to VIS_{Axis}}$	The data columns can be encoded on	A column represents grade (COL) is encoded on the y-axis (VIS_{Axis}) in a visualization (Figure 1(C))
	$\mathbb{R}_{DS \to VIS_{Type}}$	Visualization types available for the dataset are	The dataset about students' grades (DS) is visualized as a box plot (VIS_{Type}) (Figure 1(C))
Intersection of 1-to-Ns	$\mathbb{R}_{SF_1 \to COL} \cap \mathbb{R}_{SF_2 \to COL} \cap \cdots \cap \mathbb{R}_{SF_n \to COL}$	Data columns with n single-column features are	A set of columns ($COL_1 \ldots COL_n$) have the same features such as column length is 50 (SF_1), column values are sorted (SF_2) and so on ($SF_3 \ldots SF_n$)
	$\mathbb{R}_{COL_1 \to DS} \cap \mathbb{R}_{COL_2 \to DS} \cap \mathbb{R}_{CF_1 \to DS} \cap \cdots \cap \mathbb{R}_{CF_m \to DS}$	Datasets including two specific data columns and a set of m cross-columns features are	A set of datasets ($DS_1 \ldots DS_n$) have the same characteristics: their columns' characteristics ($(COL_{11}, COL_{12}) \ldots (COL_{n1}, COL_{n2})$) are the same, and these datasets have the same cross-column features such as the pairwise columns have overlapping value ranges (CF_1) and so on ($CF_2 \ldots CF_m$)

types of relation. Besides 1-to-N triples, an intersection of 1-to-N triples is also generated from the knowledge graph. For example, in Figure 4 (C), a combination of two triples, i.e.,"(Has Outlier → Column B) & (Is unique → Column B)", is an instance of intersection of 1-to-N triples. The example refers to a set of data columns with both features (i.e., Has Outlier & Is unique).

4.4 Box Embedding Learning

Box Embedding Learning guides *AdaVis* to learn possible visualization choices for a dataset. As introduced in Section 2, a head entity and a relation in triples are projected onto a box embedding. For example, suppose a single-column feature exists in data columns. In that case, this condition can be regarded as a triple like *"(A single-column feature, Exists in, Some data columns)"*. As illustrated in Figure 5 (A) and (B), the single-column feature (a yellow node) is projected into a box embedding (a hollow yellow rectangle) and its tail entities (i.e., data columns have single-column features) are supposed to lie within the box embedding. Besides transforming triples into box embeddings, multiple boxes from multiple triples can be merged to create a smaller box embedding. In Figure 5 (B) and (C), for instance, the box embeddings of columns (gray rectangles) and a cross-columns feature (a hollow orange rectangle) are intersected to obtain a smaller box embedding that represents datasets. The dataset entities with those columns and the cross-column feature will be inside the dataset box embedding. The distance between the box and tail entity is defined as:

$$\text{dist}_{\text{box}}(t;b) = \text{dist}_{\text{outside}}(t;b) + \alpha \cdot \text{dist}_{\text{inside}}(t;b) + \beta \cdot b_{\text{size}}, \quad (2)$$

where b denotes the box embedding from the head and relation in a triple, and t represents an embedding of the tail entity. The $\text{dist}_{\text{box}}(t;b)$ serves as a scoring function that measures the distance in vector space between the tail entity's the embedding and the box embedding. The distance function can be decomposed into three sub-functions: $\text{dist}_{\text{outside}}$ identifies whether the tail entity t is within the box b. If t is inside the box, the score of $\text{dist}_{\text{outside}}$ is 0. Otherwise, $\text{dist}_{\text{outside}}$ returns the distance between the tail entity t and the close side of the box b. As for $\text{dist}_{\text{inside}}$, it calculates the

distance between the center of the box b and t (or the distance between the close side of the box and t if t is outside the box). The hyper-parameter $\alpha \in [0,1]$ controls the weight of $\text{dist}_{\text{inside}}$. If $\alpha = 0$, $\text{dist}_{\text{inside}}$ is nullified, causing the scoring function to solely consider the distance of the tails from the box b. Furthermore,, b_{size} is designed to control the box size in case the box size is so large that it includes irrelevant tail entities. Thus, $\beta \in [0,1]$ is a hyparameter that controls the box size. In summary, $\text{dist}_{\text{box}}(t;b)$ aims to measure how far a tail entity is from the box embedding. $\text{dist}_{\text{outside}}$, $\text{dist}_{\text{inside}}$ and b_{size} are defined as follows:

$$\text{dist}_{\text{outside}}(t;b) = ||\text{Max}(t - b_{\text{max}}, 0) + \text{Max}(b_{\text{min}} - t, 0)||_1 \quad (3)$$

$$\text{dist}_{\text{inside}}(t;b) = ||\text{Cen}(b) - \text{Min}(b_{\text{max}}, \text{Max}(b_{\text{min}}, t))||_1, \quad (4)$$

$$b_{\text{size}}(b) = ||b_{\text{max}} - b_{\text{max}}||_2, \quad (5)$$

where b_{max} and b_{min} represent endpoints of the box b, as delineated in Figure 3(b), $\text{Cen}(b)$ denotes the box's center point. b_{size} represents the box size.

In each iteration of model training, we sample a set of positive and negative triples from the training dataset. In the knowledge graph, positive triples are the correct triples, while negative triples are incorrect triples whose tail entities t do not correspond to the head and relation. For example, *(US, Has Citizen, UK)* is a negative sample where the tail entity (*UK*) is not the answer to the head entity (*US*) and relation (*Has Citizen*). We generate k negative samples for a positive triple. The positive and negative samples constitute a minibatch of training samples. In this minibatch, *AdaVis* is updated by the calculated loss. The loss function is defined as follows, according to [21]:

$$L = -\log \sigma(\gamma - \text{dist}_{\text{box}}(b;t)) - \sum_{i=1}^{k} \frac{1}{k} \log \sigma(\text{dist}_{\text{box}}(b;t_i') - \gamma), \quad (6)$$

where σ is the Sigmoid function, and γ is a fixed scalar margin. t means a positive tail entity, while t_i' refers to a negative tail entity

that should be far away from the box embedding b. The intuition behind the loss function is that the correct answer (positive tail) should lie inside the box and as close to the box center as possible, but the incorrect answer (negative tail) should be far away from the box as possible.

4.5 Model Inference

In the training step, the model learned the embeddings of entities and relations in the knowledge graph. This section clarifies how *AdaVis* uses the learned embeddings to infer axes and possible visualization types for an unseen dataset.

    AdaVis extracts single-column and cross-column features from the unseen dataset, as illustrated in Section 4.2. Our knowledge graph contains entities of these features, and the corresponding embeddings of these features have been learned. Then, these single-column features are projected to box embeddings, as shown in Figure 5 (A) and (B). *AdaVis* further obtains the box embedding of the data column from an intersection of single-column features' box embeddings. Each single-column feature represents a particular characteristic of a data column, so the intersected box embedding of single-column features represents a data column's overall characteristics. Having obtained the data column embedding, we can infer an appropriate axis for the data column. In the paper, a dataset includes two columns, and its visualizations are also two-dimensional, with one x-axis and one y-axis. In other words, one data column only corresponds to one axis. Due to this fact, we should determine which axis is best suited to this data column. To this end, we first infer the axis' box embedding (Box_{Axis}) from this data column's box embedding (Box_{COL}). Box_{COL} represents the data column, and Box_{Axis} represents the optimal axis to this data column. The axis embedding (Box_{Axis}) is obtained from the data column embedding (Box_{COL}) by a relation ($\mathbb{R}_{COL \to VIS_{Axis}}$) which specifies the transformation from a data column to its optimal axis. Since Box_{Axis} represents the optimal axis for this column and two axis entities (i.e., x, y-axis) exists in the constructed knowledge graph, we can calculate the distance between Box_{Axis} and the embeddings of these two axis entities. The distance measures the plausibility between the optimal axis and axis entities, and the axis entities are denoted by $\mathbb{E}_{VIS_{Axis}}$ (Table 1). Additionally. the distance calculation is done by Equation 2. A lower calculated score means higher plausibility. For example, if $\text{dist}_{\text{box}}(\mathbb{E}_{VIS_x}; Box_{Axis}) < \text{dist}_{\text{box}}(\mathbb{E}_{VIS_y}; Box_{Axis})$, *AdaVis* chooses x-axis for the data column because the data column's optimal axis is more plausible to the x-axis entity than the y-axis entity.

    As for inferring visualization types to a dataset, we will identify a set of visualization types that are appropriate to the dataset. Unlike the 1-to-1 mapping between a data column and an axis, a 1-to-N mapping exists between a dataset and multiple visualization types, as multiple visualization types are suitable for the same dataset (Figure 1). To infer visualization types, we first need to obtain the dataset representation in terms of box embedding. In a manner similar to data column embedding obtainment, a box embedding of the dataset (Box_{DS}) is gained by intersecting the box embeddings of its data columns and cross-column features, as shown in Figure 5 (B) and (C). From the Box_{DS}, we infer its optimal visualization types (Box_{Type}) in terms of box embedding by a relation $\mathbb{R}_{COL \to VIS_{type}}$. Since one dataset may have more than one suitable visualization type, we need to classify these suitable visualization types at once. Due to this requirement, Equation 1 is used to identify which visualization types entities (e.g., line, bar)

are inside Box_{Type}. In other words, if visualization type entities are appropriate for the dataset, they will be enclosed by the Box_{Type}.
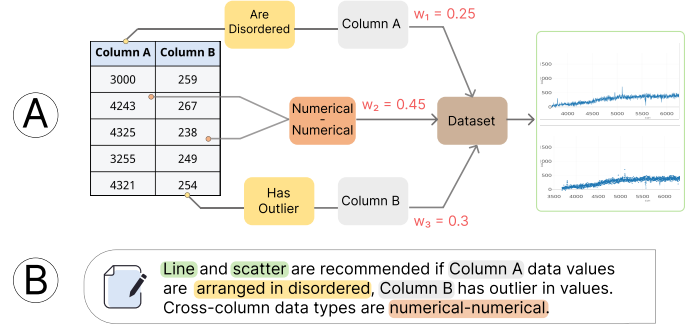

Figure 6. An illustration of explanation generation for a visualization type recommendation. (A) a visualization of three paths with quantified importance for the recommendation result. Each path represents a single-column feature (yellow) or cross-column feature (orange). Their quantified importance in the recommendation result is denoted by w_i. The w_i are obtained by normalizing their attention scores. Only parts of the paths are shown. (B) an explanation for the recommendation. The features are filled into the explanation template's slot.

4.6 Explanation Generation

In this section, we will illustrate how *AdaVis* provides fine-grained explanations for the recommendation of a specific dataset. As mentioned in Section 4.5, the procedure of *AdaVis* inference is sequential and follows an inference path, such as {Are disordered} → {Column A} → {Dataset} → {Line, Scatter} (Figure 6 (A)). The inference path indicates: ***If** data values are* ***{arranged in disorder}*** *in* ***{Column A}***, ***then** the* ***{Dataset}*** *can be visualized as* ***{Line, Scatter}***. As an inference path from a single-column feature (e.g., the column values are not ordered) or a cross-column feature (e.g., two data column's data types are both numerical) to the dataset, this path contributes to the prediction of the visualization type. To quantify each path's importance to the inference result, *AdaVis* leverages the attention mechanism in the inference paths [51]. We input box embeddings of single or cross-column features into a fully-connected neural network (i.e., MLP) and obtain output values as feature attention scores. The path's importance to the inference result is represented by its attention score, as shown in Figure 6 (A). With the quantified importance, we can reversely trace the important inference paths and reach specific single-column and cross-column feature entities.

    We have developed a rule-based template to automatically translate important features into natural language sentences. We designed an explanation template. The template is a pre-defined sentence with empty slots: "[*VIS*] is recommended if [*Column*] has [*Single-column features*], and Cross-column (i.e., the relationship between two columns) has [*Cross-column features*]". *AdaVis* recommends visualization results for the dataset and determines features that are critical to the final visualization recommendation. For example, Figure 6 (B) provides an example of the template instance where the crucial features (color-filled texts) replace the slots. Both the recommended visualization types and important features are filled in on the template. In particular, according to the principles advised by Yuan *et al.* [52], we propose detailed rules for our template to enhance the interpretability of the resulting explanation for humans. (1) We limit an explanation's maximum number of features to four to prevent cognitive overload. Also, to ensure the features in the explanation are not trivial to the recommendation result, we filter out unimportant features based

on their importance scores which are calculated by the attention score. (2) We also empirically filter out some features which are statistically informative but could confuse users, such as Moment 9, Gini coefficient, and Kurtosis. (3) We divide numerical features' semantics into several degrees based on their discretized intervals. For example, if a single-column feature such as *column length* is discretized into two intervals by MDLP, and *"a data column length is equal to 5"* is within the lower interval, the column's length will be regarded as short. (4) We avoid including categorical features that have a negative value as they are not relatable to an understandable concept for users. For instance, *"There is no linear regression"* is a negative feature value and does not make sense to users.

5 EVALUATION

To demonstrate the effectiveness of *AdaVis*, we extensively evaluated *AdaVis* with quantitative comparisons, case studies, and expert interviews. This section introduces the setup of our experiments (Section 5.1) and the results (Sections 5.2 and 5.3) in detail.

5.1 Experiment Setup

This section introduces the dataset used for evaluation and the model settings in our experiments.

Corpus. We have used the large-scale corpus of visualization-dataset pairs introduced in VizML [9] to evaluate the effectiveness of *AdaVis*. The VizML corpus is crawled from Plotly Chart Studio [3]. In the corpus, each visualization-dataset pair contains one dataset and the corresponding visualization created by users.

We first filtered all visualization-dataset pairs with the dataset consisting of two data columns from the corpus. Then, we retained four types of visualizations, i.e., bar charts, scatter plots, line charts, and box plots, since they are commonly used in Plotly [49] and are standard chart types. Our final dataset consists of approximately 30000 dataset-visualization pairs, which are further randomly divided into training and testing sets by a ratio of 2:1.

5.2 Quantitative Evaluation

We conducted experiments to evaluate *AdaVis* quantitatively from three perspectives: single-class visualization recommendation (i.e., an axis and a visualization type), multiple-class visualization recommendation (i.e., multiple visualization types), and the validity of cross-column features.

5.2.1 Single-class Visualization Recommendation

We conducted an experiment to assess our method. As mentioned in Section 4.1, a visualization consists of axes and a visualization type, so we evaluate *AdaVis* by carrying out two tasks: (1) axis recommendation for a data column; (2) visualization type recommendation for a dataset. Our corpus was collected from Plotly Chart Studio, where users typically create one visualization for each dataset. Therefore, the visualization choice is single-class in the experiment. To be specific, our experiment took one axis and one visualization type as the ground truth for each data column and dataset, respectively.

Baseline Models. We compare *AdaVis* with three baseline models: KG4Vis [18], GQE [45] and Decision Tree. Among them, KG4Vis [18] is most relevant to our approach, as it also employs

3. https://plot.ly/online-chartmaker/

a knowledge graph for recommending visualization choices and further providing explanations for its recommendations. The scores attributed by KG4Vis to each visualization choice were derived by taking an average of the inference results across all columns of a dataset. Besides KG4Vis, we also compare *AdaVis* to two other models: GQE [45] and the Decision Tree approach. GQE is a widely used model in knowledge-graph recommendation-based tasks [53], [54], [55], which makes it a suitable baseline to be compared with our approach. The Decision Tree model is essentially an explainable ML model and has also been employed in visualization recommendation [9], [31]. The models in our experiment then assign a score to each visualization option and rank them in descending order. For this, *AdaVis* employs Equation 2 to calculate the score.

Metrics. We applied two widely used metrics to evaluate the performance of our method comprehensively: Mean Rank (MR) and Hit@2 [24]. MR represents the average ranking of the correct visualization choices (the lower the MR score, the better performance), and Hits@2 represents the proportion of correct visualization choices that rank in the top two inference results (the higher the Hits@2 score, the better performance). Since the axis is either the x-axis or the y-axis, we use accuracy to evaluate its binary prediction (the higher the accuracy score, the better performance).

Result and Analysis. Table 2 shows that *AdaVis* outperforms the baseline models in recommending appropriate visualization types, underscoring the effectiveness of *AdaVis*. A contributing factor to *AdaVis*'s performance is its use of cross-column features and the intersection of box embeddings which can effectively model the intersection of all features, which extracts critical insights from both single-column and cross-column features. These insights are subsequently employed to recommend suitable visualization types. As for the axis prediction, Decision Tree marginally surpasses *AdaVis*. However, it is important to note that retraining Decision Tree for different tasks needs an additional computational burden. In contrast, *AdaVis* is trained only once for both tasks.

Table 2
The table displays the quantitative result regarding visualization type and axis recommendation among *AdaVis* and baseline models.

	Axis	Visualization Types	
	Accuracy	MR	Hits@2
AdaVis	0.8536	**1.626**	**0.8421**
GQE	0.8487	1.884	0.7268
KG4Vis	0.7579	1.736	0.8111
Decision Tree	**0.8795**	1.893	0.7189

Table 3
The table displays *AdaVis* recommendation effectiveness for multiple answers.

	Visualization Types (Adaptability)					
	Two Choices			Three Choices		
	Recall	Precision	F1	Recall	Precision	F1
AdaVis	**0.6084**	0.8259	**0.6621**	**0.6147**	0.7943	**0.6596**
GQE	0.4002	0.8005	0.5337	0.2648	0.7743	0.3972
KG4Vis	0.304	0.608	0.406	0.3333	**1.0**	0.5
Decision Tree	0.4889	**0.9778**	0.6519	0.3333	**1.0**	0.5

5.2.2 Multiple-Class Visualization Recommendation

Since each dataset has only the correct visualization type in the previous experiment, the above experiment results fail to

Table 4
The table displays an evaluation of cross-column features' effectiveness. Results are compared between models with and without cross-column features.

| | Visualization Types (Cross Features) | | | |
| | MR | | Hits@2 | |
	With	Without	With	Without
AdaVis	**1.626**	1.688	**0.8421**	0.8298
GQE	**1.884**	2	**0.7268**	0.7033
Decision Tree	**1.893**	1.9086	**0.7189**	0.7133

demonstrate the adaptability of *AdaVis*. Thus, we further conduct experiments in this subsection to evaluate the adaptability of *AdaVis* in recommending multiple types of visualizations correctly. A new test set is necessary, where each dataset has more than one correct visualization type. Since the continuous features of datasets have been transformed into categorical by discretization (Section 4.3), we grouped datasets with the same feature values. The datasets with the same single-column and cross-column features were regarded as a group whose visualization types were interchangeable. In other words, if a dataset is within a group and the group has multiple visualization types, these types will be considered the ground truth for the dataset. According to observation, a group of datasets with the same feature values are seldom visualized using four visualization types. Hence, we tested *AdaVis* adaptability with the groups of datasets whose visualization types are two and three. We sampled 406 test datasets with two visualization types and sampled 141 test datasets with three visualization types.

Baseline Models. The purpose of the experiment in this subsection is to evaluate the performance of *AdaVis* in adaptively recommending multiple types of appropriate visualizations. Given that no existing visualization recommendation approaches are explicitly designed for such a purpose, we followed the practice of Section 5.2.1 and also used KG4Vis, GQE, and Decision Tree as the baseline methods in this experiment. *AdaVis* can suggest adaptive visualization types for an unseen dataset as long as the visualization type entities are within the inference box of *AdaVis* (Equation 1). The baseline methods were set to recommend the visualization type with the highest prediction score, which is also common practice when they are used in real applications.

Metrics. For the adaptability experiment, we identify the recommended multiple visualization types based on whether the visualization type entities are inside the box of model inference (Equation 1). Since there are more than one ground truth visualizations for each test dataset, we used Recall, Precision, and F1 as the metrics in the adaptability experiment [56]. Recall can evaluate the model's adaptability. A high Recall score means the model can recommend more visualization types suited for a dataset. Precision evaluates the consistency between the model's recommendation results and the ground truth of the dataset. High precision indicates that models are well aligned with users' design preferences. F1 offers a comprehensive measurement that considers both Recall and Precision.

Result and Analysis. Table 3 shows that *AdaVis* consistently outperforms the other models in both two and three choices scenarios as indicated by the highest F1 scores, signifying that it effectively balances precision and recall. *AdaVis* also stands out in terms of recall, suggesting that it can recommend a broad range of suitable visualization types and shows excellent adaptability. In the scenarios with three choices, *AdaVis*'s performance surpasses

all baseline models, further reinforcing its adaptability. Despite some models achieving higher precision in certain scenarios, their lower overall F1 scores imply a lack of diversity in their recommendations. In contrast, *AdaVis* maintains high precision across all scenarios, indicating that its recommendations align well with user selections and can offer a wider range of suitable visualization recommendations.

5.2.3 Ablation Study about Cross-column Features

In this ablation study, we evaluated the effects of cross-column features. Given the crucial role of meaningful relationships between data columns in determining visualization type [27], [57], [58], *AdaVis* incorporates cross-column features. To verify the significance of our cross-column features, we conducted an ablation experiment. It involves removing the cross-column features from *AdaVis* and baseline models, and then assessing whether the recommended visualizations still align well with human users' visualization choices.

Baseline Models & Metrics. To evaluate the cross-column features' effect, we used the GQE and Decision Tree as the baseline models. We did not consider the KG4Vis as it does not use the cross-column features. Additionally, the metrics used in the experiment are MR and Hits@2.

Result and Analysis. Table 4 displays the effect of cross-column features on the performance of both *AdaVis* and the baseline models in recommending visualization types. In each model, i.e., *AdaVis*, GQE, and Decision Tree, the incorporation of cross-column features resulted in enhanced performance in terms of MR and Hits@2 scores. This is evident when comparing these scores with their counterparts obtained when cross-column features were not used. This trend underscores the importance of integrating cross-column features in the process of visualization recommendation. It suggests that considering the interrelations between columns (i.e., cross-column features), rather than treating each column in isolation, contributes to more effective and relevant visualization recommendations.

5.3 Qualitative Evaluation

To extensively assess *AdaVis*'s adaptability and understand why these charts are recommended, we further conducted case studies and user interviews to examine recommended visualizations and associated explanations.

5.3.1 Adaptability and Explanation

Figure 7 displays the visualization recommendation results for four datasets by *AdaVis*. As introduced in Section 4.6, *AdaVis* offers explanations for the recommendation results. These explanations highlight the features important to the recommendation results in the understandable language. The following paragraphs describe multiple recommendations for different sets.

Figure 7 (**A₁**) and (**A₂**) show that there are two types of visualizations recommended for a dataset. Figure 7 (**A₃**) explains the recommendation reason. The two columns' data types are numerical and datetime (values are related to the date or time). Additionally, in the y-axis, the numerical values are not arranged in an orderly manner (e.g., the values of the columns are arranged in increasing or decreasing order). This implies y-axis values are fluctuating. Therefore, the bar and line charts are appropriate for the dataset. The explanation is consistent with existing visualization guidances. For example, Show Me [13] concludes that a bar chart
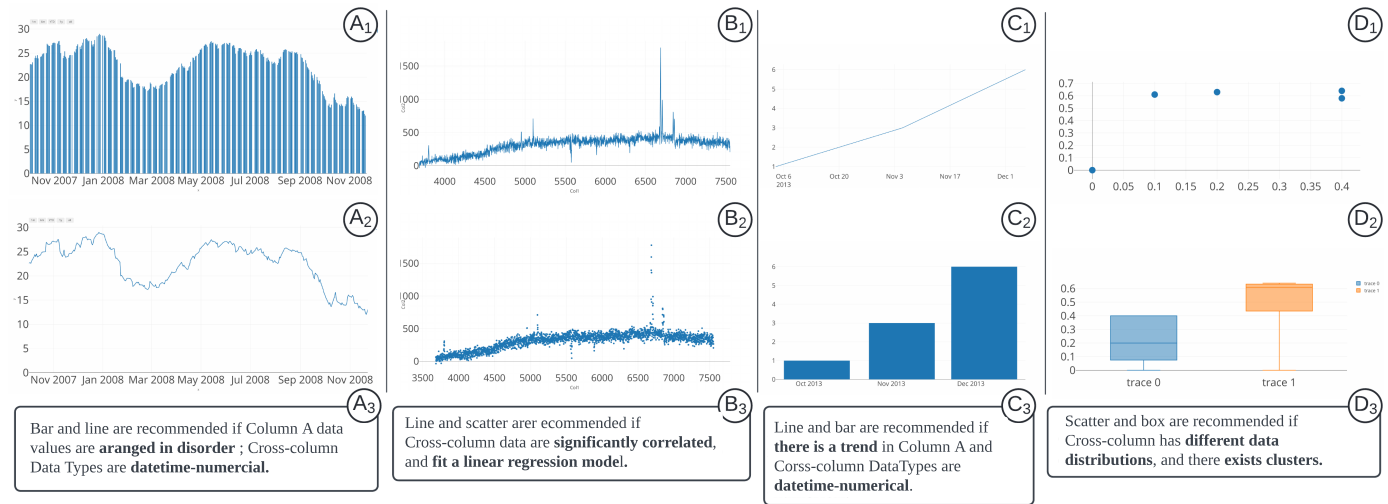
Figure 7. The present figures pairs of visualization recommendations for four different datasets. Visualizations in the same column are from the same dataset. The recommendation results are explained at the bottom of each column. Due to the space limitation, we only describe the top two features in our explanations to illustrate our recommendations.

can be used with two columns of data, one for categorical (datetime can be regarded as categorical value) and another for numerical columns. Similarly, Munzner [59] indicates that a line chart is often used to show temporal trends.

Figure 7 B_1 and B_2 present a line chart and a scatter plot as recommended visualizations for a dataset. Figure 7 B_3 shows that *AdaVis* identifies linear correlation among data columns and further recommends a line chart and a scatter plot to visualize the given dataset. The explanation is also supported by previous work. According to Cui *et al.* [48], a scatter plot is appropriate if there is a correlation between two columns of the dataset, and line charts are recommended if the dataset fits a linear regression model with a low estimated error.

Figure 7 C_1 and C_2 show that a line chart and a bar chart are recommended for a dataset by *AdaVis*. Figure 7 C_3 gives the explanations for these two recommendation results: since there is a monotonic trend in the dataset, it is suitable to visualize the dataset using a line chart because the line chart are commonly used to show the trend [59]; a bar chart is also recommended for the dataset because it contains categorical and numerical data [13].

Figure 7 D_1 and D_2 show that a scatter plot and a box plot are recommended to visualize a dataset. Figure 7 D_3 reveals important dataset features that led to these recommendation results, i.e., various data distributions and the existence of different clusters. These explanations also align well with the observations in prior studies. For example, Cui *et al.* pointed out that a scatter plot is preferred if several clusters exist in the dataset. Munzner [59] mentioned that a box plot is considered to be appropriate for a dataset with different data distribution across different columns.

These examples show that our explanations for the visualization recommendation align well with the guidelines in prior studies. It confirms the correctness of our visualization recommendation and its adaptability to satisfy the requirements of different datasets.

5.3.2 Cross-column Features

We compared recommendations generated by *AdaVis* with and without cross-column features to demonstrate their necessity. Several recommended visualizations are shown in Figure 8, where the visualizations on the left are recommended by considering

Figure 8. The figure displays pairs of visualization recommendations for four different datasets. A pair of visualization at each row comes from the same dataset. Visualizations recommended by a model with cross-column features are on the left side; Recommendations from the model without cross-column features are on the right side.

cross-column features, and those on the right are recommended without considering cross-column features. To explore how cross-column features lead to different recommendations, we studied the explanations regarding cross-column features.

Figure 8 A_1 and A_2 show that for the same dataset, *AdaVis* with cross-column features recommended scatter plot. In contrast, *AdaVis* without cross-column features recommended a bar chart. According to the model's explanation, *"there are clusters between two data columns and columns' data types are numerical-numerical"*. Since apparent clusters among data columns and two data columns are both numerical, the scatter plot is appropriate. In contrast, the bar chart is not suitable for a dataset consisting of two numerical data columns [13]. Therefore, the inappropriate visualization type is recommended because it does consider the cross-column features.

As shown in Figure 8 ((B_1), (B_2)), *AdaVis* with cross-column features recommended a line chart rather than a bar chart because *"two column data values are significantly correlated, and data types are numerical-numerical"*. The study by Saket *et al.* [60] indicates that a line chart is suitable for finding correlation. *AdaVis* with cross-column features capture the critical interrelationships between data columns in a dataset and thus recommend a line chart.

Figure 8 (C_1) and (C_2) present a box plot recommended by *AdaVis* with cross-column features and line chart without cross-column features. Based on the corresponding explanation, we recognized that *"Cross-column has a different distribution"*. That important cross-column feature led to the box plot recommendation.

Figure 8 (D_1) and (D_2) show a bar chart and a box plot, respectively. Based on the explanation, the important cross-column feature in the recommendation is that *"two column data types are categorical-numerical"*, and thus recommended a bar chart rather than a box plot. As established by Mackinaly *et al.* [13], a bar chart is well-suited for visualizing data involving two columns with categorical and numerical values. Therefore, a bar chart is an appropriate choice for visualizing the dataset. Additionally, this case illustrates that without cross-column features, *AdaVis* does not take the data type of two columns into account, leading to a box plot recommendation that violates the visualization guidance.

These cases demonstrate the importance of cross-column features in visualization recommendations. According to our observations, cross-column features can help *AdaVis* to exclude recommendations that violate visual design rules restricted by cross-column features. Cross-column features can reveal interrelationships between columns. Therefore, it is necessary to consider cross-column features in recommending appropriate visualizations.

5.3.3 User Interview and Feedback

Our quantitative experiments above demonstrate the effectiveness of *AdaVis* in adaptively recommending multiple appropriate visualizations, but it is also crucial to ask actual users to evaluate the recommended visualizations as well as the natural language explanations provided by *AdaVis*. Thus, we conducted user interviews with two distinct tasks, where each is designed to evaluate one aspect of *AdaVis*: the adaptability of our recommendations and the clarity of the generated explanations:

Task 1 **Recommendation Adaptability Assessment.** Participants were asked to view the recommended visualizations for ten randomly sampled datasets with multiple recommended visualization options. Provided with the tabular dataset, participants needed to assess and provide feedback on how well each recommended visualization presented the original tabular data of the dataset.

Task 2 **Explanation Clarity Assessment.** Participants were provided with explanations for the recommendations of ten randomly sampled datasets. They were then asked to evaluate whether the explanations helped them understand the recommendation results. Feedback was requested on the clarity of the explanations.

For the user interviews, we recruited 12 participants, all of whom actively use data visualization tools but have varying degrees of expertise in the field. This allowed us to conduct a thorough evaluation of *AdaVis* across a range of user experiences. For our analysis, we categorized them into two groups. The first group (E1-E6) is composed of participants who have demonstrated a high level of expertise in data visualization, evidenced by their

contributions to at least one scientific publication in the field. The second group (C1-C6) includes participants who regularly use data analysis tools, such as Excel, and have a fundamental understanding of data science. The diversity of participants' background allowed us to assess the effectiveness of *AdaVis* from different perspectives. Throughout the interview, we encouraged participants to express their thoughts and feedback in a think-aloud manner.

After finishing the interviews, we analyzed all the feedback from participants and also categorized participant feedback into two groups accordingly. We then conducted a thematic analysis [61] within each group to identify recurrently-raised issues. To highlight the differences between the two groups' feedback, we conducted a cross-comparison of these issues. Consequently, our analysis of the feedback revealed both convergences and divergences from the perspectives of data visualization experts and common users, which are organized and presented as follows:

Recommendations of Multiple Visualization Types. Overall, all participants found that most of the visualization recommendations by *AdaVis* are appropriate for the given datasets. For instance, E3 endorsed the variety in our recommendations: *"It is reasonable to recommend these multiple types of visualizations for the same dataset. For instance, when examining a trend or investigating correlations and distributions within a dataset, line chart, bar chart, and scatter plot can all be used to visualize the same dataset"*. However, we also observed a discrepancy between these two participant groups. The second group's (C1-C6) acceptance rate of visualization recommendations seems to be influenced by their existing knowledge of data visualization, while data visualization experts are not. For instance, C1 has never used a box plot before, and she got confused when she was presented with a box plot as the visualization recommendations. Similarly, C6 disagreed with some of the recommended line charts for datasets without a column being the time variable, as she insists that line charts should be predominantly used for time-series-related data. A significant suggestion from both groups was to further incorporate the users' analytical tasks when recommending appropriate types of visualizations for a given dataset.

Recommendation Explanations. The clarity of our explanations was confirmed by all the participants regardless of their familiarity with data visualizations. They pointed out that these explanations can help them effectively and conveniently understand why specific visualizations were recommended for a given dataset. For example, C6 mentioned that the explanations enhanced her comprehension of the recommended visualization, as they highlighted the specific features that drove the recommendations. Nevertheless, we observed that a user's knowledge level of visualization significantly influences their requirements for explanations. Users less familiar with data visualization may need more detailed and contextual explanations of the recommended visualization. For example, C4 suggested that the explanation should be task-oriented, displaying a specific scenario, and C2 expressed a desire for justifications in explanation when his unfamiliar visualizations are recommended. Conversely, users with more advanced knowledge, like E2, might prefer an explanation that focuses on the characteristics of a dataset. This observation underscores the importance of tailoring explanations to the knowledge level and needs of individual users to facilitate their understanding and acceptance of the recommended visualizations. Also, participants have provided insightful suggestions for further enhancing *AdaVis*. For instance, a common suggestion from both groups of participants was that our explanation should also illustrate why certain types

of visualizations were not recommended beyond only explaining why certain visualizations were recommended. More specifically, E1 advised that *AdaVis* could incorporate a *What-if* functionality, enabling users to discern which features are most influential to recommendation results. Moreover, some common users (C6, C5) pointed out that certain terms used in the explanation, such as "disorder", should be presented in a more intuitive manner.

6 DISCUSSION

6.1 Lessons

Explanability. Our explanation for visualization recommendations considers both feature importance and intuitiveness. 120 features are used in our approach to comprehensively model the characteristics of the input dataset. However, an increasing number of dataset features can also result in the difficulty of providing intuitive explanations for the visualization recommendation from the perspective of dataset features. To strike a good balance between visualization recommendation performance and explainability, we integrated an attention mechanism in *AdaVis* (Section 4.6) that can identify critical features for the final visualization recommendation result. Our explanation is built upon the most important two or three features. For feature intuitiveness, some dataset features are important for the final visualization recommendation result, but their meanings are difficult to interpret, especially for statistical data features (i.e., the entropy is high, Kolmogorov–Smirnov test result is significant). Given that target audiences comprise common users, these perplexing features were avoided in the final explanations.

Trade-off in Model Training Strategy. *AdaVis* performs two kinds of recommendation tasks: axis and visualization type recommendation. Since single-column features are required in both axes and visualization type prediction, the embeddings of the single-column features are influenced by two tasks about axis and visualization types recommendation, when we combine the two tasks for the model to learn; the multi-task learning can introduce noise because visualization type recommendation visualization is unrelated to axes prediction. To investigate the effect, we conducted a control experiment on multi-task and single-task learning; the model was doing multi-task learning while two separate models were trained for each task in the control group. According to the experiment result, training different models for each task led to a few points of improvement. However, the improvement came at the cost of double the computation time and storage, because it needs to train two individual models to do axis and visualization type recommendation task, respectively.

Feature Importance. *AdaVis* defines a comprehensive set of features, including 80 single-column features and 40 cross-column features, to capture the diverse characteristics of input datasets. A feature importance analysis revealed that some single-column features, especially those related to column names and data column statistical properties, are particularly impactful. Among these, *data column length* emerges as a highly influential feature. The significance of data column length can be attributed to its role in reflecting data density, which in turn informs the choice of visualization type. For instance, for a high-density dataset, a line chart may be more suitable than a bar chart or scatter plot, as it represents data points in a less cluttered and more understandable way. Also, our analysis highlights the importance of *digits in column name*. This feature carries semantic information about the data column, which potentially indicates users' design logic for visualization. For example, a column named "Year2017" can imply

the need to visualize a trend over time, while column names such as "Method1" or "Method2" only reveal the need for a comparison.

6.2 Limitations

Corpus. We utilize the dataset-visualization pairs uploaded by Plotly users as the ground truth, and most visualization choices of them align well with general visualization design guidelines. However, there are also a small number of problematic visualization choices for the input datasets, which may have a negative impact on the performance of *AdaVis*. To further bolster the performance of *AdaVis*, it is important to expand our corpus with more high-quality dataset-visualization pairs. For example, we can try to collect more data visualization examples created by experienced visualization experts from professional visualization blogs or forums like Observable[4].

Visualization Choices. As an initial step towards adaptive and explainable visualization recommendation, *AdaVis* is applied to the widely-used standard charts in this paper, like line charts, bar charts, scatter plots, and box plots, and it does not encompass all types of visualizations. Such a choice originates from both the popularity of these visualizations [9] and the fact that other visualizations are scarce in the Plotly corpus. Also, given that these standard charts have an emphasis on the axes' visual encodings [9], *AdaVis* mainly focuses on recommending appropriate types of data visualizations and the x/y axes. However, with new dataset-visualization corpus of other types of visualizations, *AdaVis* can be easily extended to work for new types of visualizations and other detailed visualization encodings like color schemes.

User-centric Recommendation. *AdaVis* effectively maps datasets to visualizations but lacks an explicit consideration of users' specific intents, such as their analytical tasks or preferences. As indicated by user feedback (Section 5.3.3), it will be interesting to further incorporate user intent in visualization recommendations, which can ensure that the recommended visualizations align with the user's specific needs. In this paper, we demonstrate the effectiveness of *AdaVis* by using datasets with two columns, but *AdaVis* can also recommend appropriate visualizations for datasets with more than two columns. It can be achieved by extracting cross-column features from every possible combinations of two columns in the dataset, and then finding the intersection of these cross-column box embeddings, which indicate the dataset's characteristics and can be further used to derive the appropriate the visualization choices. The possible issue of extending *AdaVis* to datasets with over two columns is that the exhaustive search of every possible combination of two columns in the dataset can be time-consuming, which can be mitigated by further considering user intent to narrow down the search space of pairwise column combination in the visualization recommendation process. In addition, some terminologies used in the natural language explanations by *AdaVis* may not be easily understood by all users. For instance, technical terms like "a linear regression model" are obvious for machine learning practitioners but can be perplexing for laypersons. It will be helpful to further incorporate more straightforward explanations into *AdaVis*, making it more accessible to a broader range of audiences.

Training Time. The training time of *AdaVis* is prolonged due to a large number of extracted features and corpus. *AdaVis* extracts many features from a dataset. These features enable *AdaVis* to comprehensively model the dataset characteristics and further increase the generability of *AdaVis*, which can recommend

4. https://observablehq.com/

appropriate visualization types for diverse datasets. However, the large number of features also increases model complexity and thus enlarges the model size. In addition, to better learn the complex mapping from datasets to visualization types, *AdaVis* is trained on a large corpus. Feature importance analysis can be used to identify unimportant features and reduce the feature number in the model.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose *AdaVis*, an adaptive and explainable approach for visualization recommendation. Given a dataset, *AdaVis* can adaptively recommend multiple appropriate visualization choices and provide detailed explanations for the recommendation result. Our approach consists of four modules: feature extraction, knowledge graph construction, model training, and inference. It first extracts the individual column's features and the interrelationship among data features, data columns and visualization choices. With these features and interrelationships, a knowledge graph is constructed to model them. The box embeddings of entities and relations in the knowledge graph can also be learned. With these learned box embeddings, an inference module can adaptively recommend multiple visualizations for an unseen dataset and provide natural language explanations for the recommendations. Quantitative evaluations were conducted to evaluate the effectiveness of visualization choice recommendation, the adaptiveness of *AdaVis* and benefit of cross-column features. To assess the adaptability of *AdaVis*, we conducted qualitative evaluations such as case studies and user interviews to further demonstrate it.

In future work, we will collect more diverse dataset-visualization pairs and extend *AdaVis* to recommend more different types of data visualizations (i.e., heatmap, histogram) in an adaptive and explainable manner. Also, it is interesting to investigate how user intent, such as preferences and analytical tasks, can be integrated into *AdaVis* to further improve its efficiency and effectiveness in adaptive and explainable visualization recommendations.

REFERENCES

[1] M. Zhou, Q. Li, Y. Li, S. Han, and D. Zhang, "Table2charts: Learning shared representations for recommending charts on multi-dimensional data," pp. 2389–2399, 2020.

[2] M. O. Ward, G. Grinstein, and D. Keim, Interactive data visualization: foundations, techniques, and applications.   CRC Press, 2010.

[3] H. Wickham, "A layered grammar of graphics," Journal of Computational and Graphical Statistics, vol. 19, no. 1, pp. 3–28, 2010.

[4] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer, "Reactive vega: A streaming dataflow architecture for declarative interactive visualization," IEEE Transactions on Visualization and Computer Graphics, vol. 22, no. 1, pp. 659–668, 2015.

[5] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 12, pp. 2301–2309, 2011.

[6] J. Heer, S. K. Card, and J. A. Landay, "Prefuse: a toolkit for interactive information visualization," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2005, pp. 421–430.

[7] V. Dibia and Ç. Demiralp, "Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks," IEEE Computer Graphics and Applications, vol. 39, no. 5, pp. 33–46, 2019.

[8] Y. Wang, Z. Jin, Q. Wang, W. Cui, T. Ma, and H. Qu, "Deepdrawing: A deep learning approach to graph drawing," IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 1, pp. 676–686, 2019.

[9] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo, "Vizml: A machine learning approach to visualization recommendation," in Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 2019, pp. 1–12.

[10] Y. Li, Y. Qi, Y. Shi, Q. Chen, N. Cao, and S. Chen, "Diverse interaction recommendation for public users exploring multi-view visualization using deep learning," IEEE Transactions on Visualization and Computer Graphics, vol. 29, no. 1, pp. 95–105, 2022.

[11] Z. Zeng, P. Moh, F. Du, J. Hoffswell, T. Y. Lee, S. Malik, E. Koh, and L. Battle, "An evaluation-focused framework for visualization recommendation algorithms," IEEE Transactions on Visualization and Computer Graphics, vol. PP, pp. 1–1, 2021.

[12] J. D. Mackinlay, "Automating the design of graphical presentations of relational information," ACM Transactions on Graphics, vol. 5, no. 2, pp. 110–141, 1986.

[13] J. Mackinlay, P. Hanrahan, and C. Stolte, "Show me: Automatic presentation for visual analysis," IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 6, pp. 1137–1144, 2007.

[14] M. Vartak, A. Parameswaran, N. Polyzotis, and S. R. Madden, "Seedb: automatically generating query visualizations," 2014.

[15] S. Zhu, G. Sun, Q. Jiang, M. Zha, and R. Liang, "A survey on automatic infographics and visualization recommendations," Visual Informatics, vol. 4, no. 3, pp. 24–40, 2020.

[16] Q. Wang, Z. Chen, Y. Wang, and H. Qu, "A survey on ml4vis: Applying machinelearning advances to data visualization," IEEE Transactions on Visualization and Computer Graphics, 2021.

[17] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang, and H. Qu, "Ai4vis: Survey on artificial intelligence approaches for data visualization," IEEE Transactions on Visualization and Computer Graphics, 2021.

[18] H. Li, Y. Wang, S. Zhang, Y. Song, and H. Qu, "Kg4vis: A knowledge graph-based approach for visualization recommendation," IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 195–205, 2021.

[19] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, methods, and applications in interpretable machine learning," Proceedings of the National Academy of Sciences, vol. 116, no. 44, pp. 22 071–22 080, 2019.

[20] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in Proceedings of the 27th Annual Conference on Neural Information Processing Systems 2013, 2013, pp. 2787–2795.

[21] H. Ren, W. Hu, and J. Leskovec, "Query2box: Reasoning over knowledge graphs in vector space using box embeddings," arXiv preprint arXiv:2002.05969, 2020.

[22] W. Chen and K. Shi, "Multi-scale attention convolutional neural network for time series classification," Neural Networks, vol. 136, pp. 126–140, 2021.

[23] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 12, pp. 2724–2743, 2017.

[24] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," Advances in Neural Information Processing Systems, vol. 26, 2013.

[25] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, "Modeling relation paths for representation learning of knowledge bases," arXiv preprint arXiv:1506.00379, 2015.

[26] L. Vilnis, X. Li, S. Murty, and A. McCallum, "Probabilistic embedding of knowledge graphs with box lattice measures," arXiv preprint arXiv:1805.06627, 2018.

[27] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer, "Voyager: Exploratory analysis via faceted browsing of visualization recommendations," IEEE Transactions on Visualization and Computer Graphics, vol. 22, no. 1, pp. 649–658, 2015.

[28] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer, "Voyager 2: Augmenting visual analysis with partial view specifications," in Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017, pp. 2648–2659.

[29] Ç. Demiralp, P. J. Haas, S. Parthasarathy, and T. Pedapati, "Foresight: Recommending visual insights," arXiv preprint arXiv:1707.03877, 2017.

[30] A. Key, B. Howe, D. Perry, and C. Aragon, "Vizdeck: self-organizing dashboards for visual analytics," in Proceedings of the 2012 ACM

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3316469

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. XX, NO. XX, XX 2018   14

SIGMOD International Conference on Management of Data, 2012, pp. 681–684.

[31] Y. Luo, X. Qin, N. Tang, and G. Li, "Deepeye: Towards automatic data visualization," in Proceedings of 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE, 2018, pp. 101–112.

[32] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 89–96.

[33] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer, "Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco," IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 1, pp. 438–448, 2018.

[34] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," IEEE Transactions on Neural Networks and Learning Systems, 2021.

[35] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in International Conference on Machine Learning, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:1157792

[36] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," arXiv preprint arXiv:1412.6575, 2014.

[37] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28, no. 1, 2014.

[38] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, and Z. Duan, "Knowledge graph completion: A review," IEEE Access, vol. 8, pp. 192 435–192 456, 2020.

[39] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, 2015, pp. 687–696.

[40] X.-H. Li, C. C. Cao, Y. Shi, W. Bai, H. Gao, L. Qiu, C. Wang, Y. Gao, S. Zhang, X. Xue et al., "A survey of data-driven and knowledge-aware explainable ai," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 1, pp. 29–49, 2020.

[41] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, 2019, pp. 5329–5336.

[42] W. Ma, M. Zhang, Y. Cao, W. Jin, C. Wang, Y. Liu, S. Ma, and X. Ren, "Jointly learning explainable rules for recommendation with knowledge graph," in The World Wide Web Conference, 2019, pp. 1210–1221.

[43] O. Seneviratne, A. K. Das, S. Chari, N. N. Agu, S. M. Rashid, C.-H. Chen, J. P. McCusker, J. A. Hendler, and D. L. McGuinness, "Enabling trust in clinical decision support recommendations through semantics." in SeWeBMeDa@ ISWC, 2019, pp. 55–67.

[44] M. K. Sarker, N. Xie, D. Doran, M. Raymer, and P. Hitzler, "Explaining trained neural networks with semantic web technologies: First steps," arXiv preprint arXiv:1710.04324, 2017.

[45] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec, "Embedding logical queries on knowledge graphs," Advances in Neural Information Processing Systems, vol. 31, 2018.

[46] R. Ding, S. Han, Y. Xu, H. Zhang, and D. Zhang, "Quickinsights: Quick and automatic discovery of insights from multi-dimensional data," in Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 317–332.

[47] C. Harris, R. A. Rossi, S. Malik, J. Hoffswell, F. Du, T. Y. Lee, E. Koh, and H. Zhao, "Insight-centric visualization recommendation," arXiv preprint arXiv:2103.11297, 2021.

[48] Z. Cui, S. K. Badam, M. A. Yalçin, and N. Elmqvist, "Datasite: Proactive visual data exploration with computation of insight-based recommendations," Information Visualization, vol. 18, no. 2, pp. 251–267, 2019.

[49] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker, "Beagle: Automated extraction and interpretation of visualizations from the web," in Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, 2018, pp. 1–8.

[50] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An enabling technique," Data Mining and Knowledge Discovery, vol. 6, no. 4, pp. 393–423, 2002.

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

[52] J. Yuan, J. Vig, and N. Rajani, "isea: An interactive pipeline for semantic error analysis of nlp models," in 27th International Conference on Intelligent User Interfaces, 2022, pp. 878–888.

[53] H. Ren and J. Leskovec, "Beta embeddings for multi-hop logical reasoning in knowledge graphs," Advances in Neural Information Processing Systems, vol. 33, pp. 19 716–19 726, 2020.

[54] Z. Zhang, J. Wang, J. Chen, S. Ji, and F. Wu, "Cone: Cone embeddings for multi-hop reasoning over knowledge graphs," Advances in Neural Information Processing Systems, vol. 34, pp. 19 172–19 183, 2021.

[55] Z. Hu, V. Gutiérrez-Basulto, Z. Xiang, X. Li, R. Li, and J. Z. Pan, "Type-aware embeddings for multi-hop reasoning over knowledge graphs," arXiv preprint arXiv:2205.00782, 2022.

[56] S. Raschka, "An overview of general performance metrics of binary classifier systems," arXiv preprint arXiv:1410.5330, 2014.

[57] G. Wills and L. Wilkinson, "Autovis: automatic visualization," Information Visualization, vol. 9, no. 1, pp. 47–69, 2010.

[58] J. Seo and B. Shneiderman, "A rank-by-feature framework for interactive exploration of multidimensional data," Information visualization, vol. 4, no. 2, pp. 96–113, 2005.

[59] T. Munzner, Visualization Analysis and Design. CRC press, 2014.

[60] B. Saket, A. Endert, and Ç. Demiralp, "Task-based effectiveness of basic visualizations," IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 7, pp. 2505–2512, 2018.

[61] G. Guest, K. M. MacQueen, and E. E. Namey, Applied thematic analysis. sage publications, 2011.

Songheng Zhang is currently a Ph.D. candidate in the School of Computing and Information Systems at Singapore Management University. He received his double B.E. degree in Computer Science and Mathematics from the University of California, Irvine in 2021. His research interests include data visualization and visual analytics.

Yong Wang is currently an assistant professor in School of Computing and Information Systems at Singapore Management University. His research interests include data visualization, visual analytics and explainable machine learning. He obtained his Ph.D. in Computer Science from Hong Kong University of Science and Technology in 2018. He received his B.E. and M.E. from Harbin Institute of Technology and Huazhong University of Science and Technology, respectively. For more details, please refer to http://yong-wang.org.

Haotian Li is currently a Ph.D. candidate in Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). His main research interests are data visualization, visual analytics, human-computer interaction and online education. He received his BEng in Computer Engineering from HKUST. For more details, please refer to https://haotian-li.com/.

Huamin Qu is a professor in the Department of Computer Science and Engineering (CSE) at the Hong Kong University of Science and Technology (HKUST) and also the director of the interdisciplinary program office (IPO) of HKUST. He obtained a BS in Mathematics from Xi'an Jiaotong University, China, an MS and a PhD in Computer Science from the Stony Brook University. His main research interests are in visualization and human-computer interaction, with focuses on urban informatics, social network analysis, E-learning, text visualization, and explainable artificial intelligence (XAI).