VisGNN: Personalized Visualization Recommendation via Graph Neural Networks

Fayokemi Ojo
Johns Hopkins University

Ryan A. Rossi
Adobe Research

Jane Hoffswell
Adobe Research

Shunan Guo
Adobe Research

Fan Du
Adobe Research

Sungchul Kim
Adobe Research

Chang Xiao
Adobe Research

Eunyee Koh
Adobe Research

ABSTRACT

In this work, we develop a Graph Neural Network (GNN) framework for the problem of personalized visualization recommendation. The GNN-based framework first represents the large corpus of datasets and visualizations from users as a large heterogeneous graph. Then, it decomposes a visualization into its data and visual components, and then jointly models each of them as a large graph to obtain embeddings of the users, attributes (across all datasets in the corpus), and visual-configurations. From these user-specific embeddings of the attributes and visual-configurations, we can predict the probability of any visualization arising from a specific user. Finally, the experiments demonstrated the effectiveness of using graph neural networks for automatic and personalized recommendation of visualizations to specific users based on their data and visual (design choice) preferences. To the best of our knowledge, this is the first such work to develop and leverage GNNs for this problem.

KEYWORDS

Personalized Visualization Recommendation, Attribute Recommendation, Graph Neural Networks, User Modeling, Personalization

1 INTRODUCTION

Visualization recommendation systems are important for many web applications including exploratory data analysis and dashboard creation, among many others. The goal of such systems is to improve the user experience by providing a set of proper visualizations for users to efficiently and effectively find important patterns and insights from their data for decision-making, marketing, and so on. In many cases, these systems allow a user to select or upload a dataset of interest and then immediately see potentially interesting visualizations for their given dataset. However, existing work is all rule-based [40, 41], and thus unable to recommend visualizations that are personalized to specific users based on the previous visualizations that they preferred, liked, or generated.

In this work, we focus on the personalized visualization recommendation problem [26] where we leverage both implicit and explicit user feedback to automatically learn a personalized model for each user. Such a model can better recommend relevant visualizations to the user in real-time given any new unseen dataset of interest. The goal of personalized visualization recommendation is to learn individual recommendation models for each user. Then when a user selects a dataset of interest to explore, the specific model learned for that user can be applied and the top visualizations that are most relevant and related to them will be recommended.

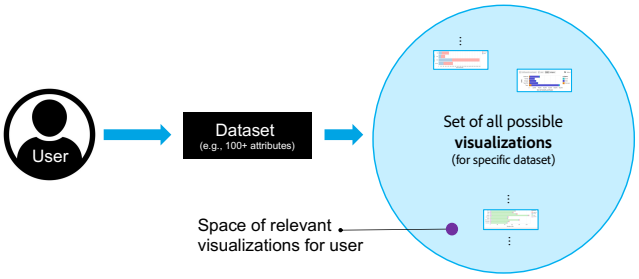

Figure 1: Given a user i and an arbitrary dataset of interest to them (selected or uploaded by the user), the space of possible visualizations to recommend are intractable. Despite this intractable exponential space of all possible visualizations that can be generated from the users dataset of interest, there is only a very small subspace of visualizations that are relevant and of interest to the user.

However, there are several challenges that make it difficult to provide effective and personalized visualization recommendations. First of all, users typically have their own datasets that are not shared by any other user. Therefore, traditional collaborative filtering approaches do not work in this setting. To mitigate this issue, we introduce a GNN-based framework that leverages the large corpus of datasets and visualizations from users as a large heterogeneous graph. Secondly, the complexity of the visualization space makes this problem even harder. Specifically, for a single dataset, the set of possible candidate visualizations to recommend to a user is exponential in the number of attributes in the dataset, possible design choices (*e.g.*, chart-types, colors, sizes, x/y, etc.), and so on, making this a fundamentally challenging problem (Figure 1). Hence, the space of visualizations for one dataset is intractable already. To make matters worse, there are tens of thousands of datasets and each dataset is often only utilized by a single user, or at most a few such users. This makes an already difficult problem even more challenging to solve since it implies a very limited

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512001

amount of feedback per dataset, despite the intractable space of visualizations. Furthermore, since a visualization consists of both the particular data attributes used as well as the corresponding design choices, the visualization is fundamentally tied to a specific dataset, and therefore, the space of visualizations for one dataset is completely disjoint from the space of visualizations from another dataset (Figure 2).

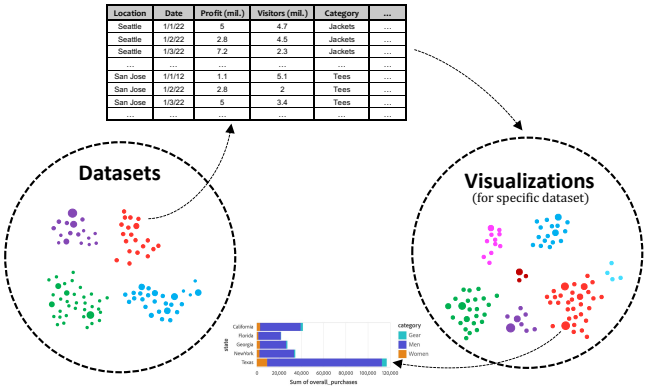

Figure 2: Space of visualizations to score is dependent on the specific dataset of interest to a user. Hence, given a different dataset of interest by some other user almost surely leads to an entirely disjoint space of visualizations to recommend. Intuitively, the space of datasets are shown on the left where each point represents a different dataset, and the color encodes the domain of the dataset. In this simple example, there are datasets from four different domains. An example of one such dataset is shown in the middle, and it includes attributes such as location, date, category, profit, visitors, and so on. From this dataset, we then show the space of visualizations that can be generated from using one or more data attributes.

In this work, we develop a Graph Neural Network (GNN) framework called VisGNN for personalized visualization recommendation. Specifically, given a set of users, their datasets of interest (where each dataset consists of a set of data-attributes), and the visualizations that users generated from those datasets, VisGNN first derives a series of graphs where nodes are users, data-attributes (from all datasets), and all possible visual-configurations from the corpus of visualizations. VisGNN encodes the interactions between users and each of the data-attributes from a visualization of interest as edges in the graph. Furthermore, VisGNN also derives edges between users and the visual-configurations that were extracted from a visualization preferred by that user, and also encodes the relationship between the visual-configuration and data-attributes used in the visualization that the visual-configuration was extracted. Every data-attribute from each dataset is also mapped to a shared low-dimensional space that enables direct comparison of the data-attributes across different datasets. We then learn initial feature vectors for every user, data-attribute, and visual-configuration, which are used as the node features in the initial layer of our GNN. Feature information is then iteratively aggregated from neighbors and the aggregated information is then encoded with the existing node

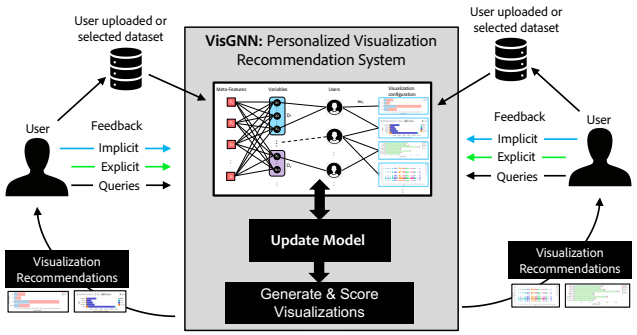

Figure 3: Overview of the web-based VisGNN Personalized Visualization Recommendation System. Users select/upload a dataset of interest using the web UI, then VisGNN updates the graph representations and user-specific models. The model learned for the specific user is then used to obtain personalized scores of the visualizations for that specific user. The top visualizations for each user and their dataset of interest are then displayed to the user (from most relevant to least). Quality of the personalized recommendations for the user improves over time as the system continuously learns from additional implicit/explicit feedback from the user on the visualizations.

representation during propagation. VisGNN effectively captures the non-linear interactions between the users, data-attributes, and visual-configurations, thereby improving the learned representations, making it possible to infer highly relevant and interesting visualizations personalized for specific users.

VisGNN derives a *user-specific visualization representation* by concatenating all the learned representations that pertain to a specific visualization, *i.e.*, representations of the user, data-attributes, and visual-configuration that make up a visualization. We then learn a non-linear function that maps the user-specific visualization representation to a positive or negative class label that encodes whether the user preferred the visualization or not. In addition to the few positive visualizations for a given user, VisGNN is also trained using many negative visualizations, which are visualizations that a specific user did not find interesting or relevant. Now, given a new potential visualization for some arbitrary user, the trained model can be used to accurately infer the probability of the visualization being interesting or useful for that user. VisGNN is also general and flexible with many interchangeable components, making it useful for many different use cases (such as personalized attribute recommendation). The proposed approach is able to effectively deal with the vast sparsity and cold-start issues that make this problem impractical using traditional methods. To the best of our knowledge, this is the first work to develop a GNN-based approach to solve this important problem.

There are a variety of different, yet complex implicit and explicit user feedback that can be leveraged by VisGNN. Since visualization recommendation systems typically show users visualizations, this is the most common type of user feedback. See Figure 3 for an overview of the web-based personalized visualization recommendation system. In particular, a user may "like" or "add a visualization

to their dashboard". These are all examples of explicit user feedback. In contrast, examples of implicit user feedback include when a user clicks or hover-over a visualization. However, feedback directly on the visualizations is not very useful for our problem due to the issues discussed in the previous paragraphs.

To overcome these issues, we decompose every visualization into two parts: the data attributes and the set of visual design choices used in the visualization, which we call a visual-configuration (Figure 4). An important detail is that the set of design choices do not include the actual data-attributes, *e.g.*, if a data attribute from some arbitrary dataset was mapped to the color or x/y-axis, then we replace the attribute name with some set of properties such as whether it is numerical, categorical, and so on (attribute data type). Decomposing the user-preferred visualization into the data attributes and visual design choices used in it, enables us to learn from user feedback. While the user feedback on the data-attributes used in the visualization does not directly transfer for visualizations created from a different dataset, the set of design choices preferred by the user does, and we can leverage it in our model to recommend better personalized visualizations for users. In many applications or tools, there also exist direct implicit and explicit user feedback on the attributes and design-choices of interest.

Through extensive experiments, we demonstrate the effectiveness and utility of our approach using a large-scale corpus of 17.4k users with 94k datasets (2.3 million attributes in total) and 32k visualizations generated from those datasets. Overall, VisGNN outperforms a variety of baseline methods. We also perform an ablation study to investigate a few other GNN-based variants of our approach. Finally, we investigate using the GNN-based approach for recommending data attributes to individual users based on their learned preferences.

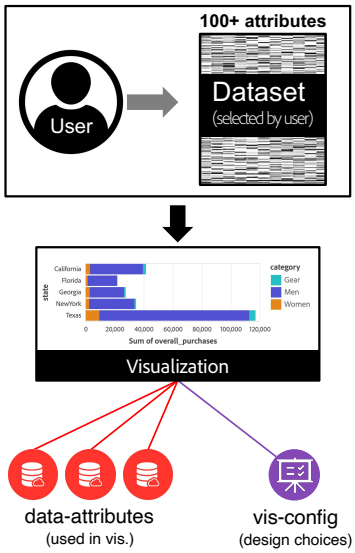

Figure 4: Visualizations are decomposed into data-attributes and design choices (visual-configuration). While visualizations are dataset dependent, visual-configurations are not. This approach enables VisGNN to learn from user feedback across thousands of entirely different datasets.

2 RELATED WORK

While there has been a lot of work on visualization recommendation, there has only been a few such works that leverage models learned from data. Furthermore, none of these works are based on graph neural networks (GNNs).

2.1 Visualization Recommendation

Traditional user-driven workflow for creating data visualizations contains stages of selecting datasets or subsets, data attributes, visualization components and interactions [36]. To lower the barriers of the required visualization knowledge and accelerate this manual process, several automatic visualization recommendation techniques have been developed to help choosing the visualization components and configuring them. Rule-based visualization recommendation systems such as Voyager [34, 41, 42], VizDeck [24], and DIVE [13] use a large set of human perceptual effectiveness metrics defined manually by domain experts to recommend appropriate visualizations that satisfy the rules [2, 6, 10, 15, 19, 20, 29, 30, 32]. Such rule-based systems do not leverage any training data for learning or user personalization. There have been a few "hybrid" approaches that combine some form of learning with manually defined rules for visualization recommendation [22], *e.g.*, Draco learns weights for rules (constraints) [22]. Recently, there has been work that focused on the end-to-end ML-based visualization recommendation problem [7, 27]. However, the above work learns a global visualization recommendation model that is agnostic of the user, and thus not able to be used for the personalized visualization recommendation problem studied in our work.

All of the existing rule-based [13, 24, 34, 41, 42], hybrid [22], and pure ML-based visualization recommendation [27] approaches are unable to recommend personalized visualizations for specific users. These approaches do not model users, but focus entirely on learning or manually defining visualization rules that capture the notion of an effective visualization [3–5, 8, 14, 16, 17, 21, 31, 35, 38, 39]. Therefore, no matter the user, the model always gives the same recommendations. One recent work called VizRec [23] studied the setting where there is a single dataset shared by all users, and therefore a single small set of visualizations that the users have explicitly liked and tagged, which are then recommended to the users based on a very simple heuristic approach. This problem is unrealistic with many impractical assumptions that are not aligned with practice. More recently, Qian *et al.* [26] introduced the problem of personalized visualization recommendation and proposed an approach capable of solving it. However, that work relied on a large dense meta-feature matrix to characterize the attributes across datasets. Further, they do not introduce nor investigate a graph neural network framework for personalized visualization recommendation.

While most research effort has been focused on recommending visualization, very few studies work towards the automation of data attribution selection, which is also a critical step of the visualization creation workflow. Some visualization recommendation systems, such as SeeDB [35], Voyager [41, 42] and DeepEye [18]

require users to specify interested data variables or query the attributes by keywords before recommending visual configurations. The selection of data attributes are closely related to personalized visualization settings, where the importance of attributes may vary according to users' roles and exploration behaviors. Therefore, most of the non-personalized visualization recommendation techniques fail to support automatic attribute recommendation. Our method addresses this problem by aggregating the information of visual configurations and data attributes when training the model, and predicts the probabilities of each data attributes to be of user's interest while making visualization recommendation.

2.2 Graph Neural Networks

GNNs have been successfully applied to a variety of important applications including neural machine translation in text [1], content-based recommendation systems [9, 11, 33, 37, 45], and human-object interaction detection [25]. We refer the reader to the survey [43, 46] for a detailed list of GNNs proposed for more an even wider range of GNN applications. Within these applications, GNNs provide state-of-the-art results for tasks related to node classification, link prediction problems, and graph classification [44]. To the best of our knowledge, our work is the first one to use GNNs for visualization recommendation.

3 APPROACH

In this section, we formally introduce our GNN framework for the personalized visualization problem called VisGNN. Given an arbitrary dataset, we recommend the top-k visualizations personalized for the given user based on the users previous visual and data preferences.

Given a visualization $\mathcal{V}$ from user i for some arbitrary dataset of interest, we decompose the visualization into the set of data attributes $\mathcal{A}$ used in the visualization and the set of visual design choices (which we also call the visual-configuration, see Figure 4). We set $A_{ij} = 1$ for all data attributes $j \in \mathcal{A}$ and $C_{ik} = 1$ for the extracted visual configuration k (which represents the complete set of design choices). We also include nodes for the other data-attributes in the user's dataset that have not yet been used in a visualization. Initially, these attribute nodes are not connected to any other node in the graph. In addition to the two graphs $\mathbf{A}$ and $\mathbf{C}$ described above, we also encode the attributes $\mathcal{A}$ used in the specific visual-configuration k using another graph $\mathbf{D}$. More specifically, $D_{jk} = 1$ for all $j \in \mathcal{A}$ and k is the visual-configuration of the visualization $\mathcal{V}$. This approach results in three graphs, encoded by the sparse adjacency matrices $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{D}$. Given these graphs, we first derive a larger unified graph as follows:

$$\mathbf{G} = \begin{bmatrix} \blacksquare & \mathbf{A} & \mathbf{C} \\ \mathbf{A}^\top & \blacksquare & \mathbf{D} \\ \mathbf{C}^\top & \mathbf{D}^\top & \blacksquare \end{bmatrix} \tag{1}$$

In Figure 5, we provide an example of the resulting unified graph shown on the right. Notably, the graph shows three users whom are connected to the visual-configurations (set of design choices) and data-attributes used in the visualizations that they provided positive user-relevant feedback. In this graph, there is no notion of a visualization, however, one can generate a valid visualization by combining one or more data-attributes with a visual-configuration.

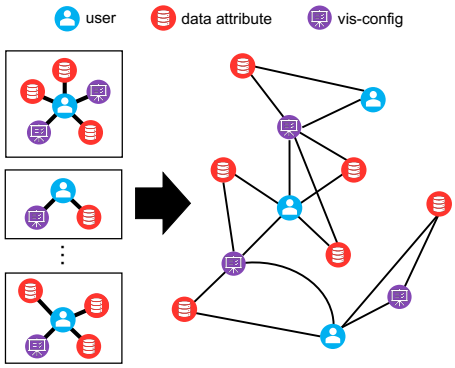

Figure 5: Every subgraph shown on the left pertains to a single dataset and user. There are many data-attributes from a users dataset of interest that are not used in a visualization. These data-attributes are not shown in the above figure for simplicity. On the right, we see the unified graph.

Then, we obtain a low-dimensional rank-d approximation of $\mathbf{G}$ denoted as $\phi(\mathbf{G})$. Given the large heterogeneous graph $\mathbf{G}$ and $\phi(\mathbf{G})$, we have a graph neural network layer of the following form:

$$\mathbf{H}^{k+1} = f(\mathbf{H}^{(k)}, \ell(\mathbf{G})) \tag{2}$$

where f is a non-linear function over $\mathbf{H}^{(k)}$ and the graph $\mathbf{G}$. For the initial GNN layer $k = 0$, we have:

$$\mathbf{H}^{1} = f(\phi(\mathbf{G}), \ell(\mathbf{G})) \tag{3}$$

where $\mathbf{H}^0 = \phi(\mathbf{G}) \in \mathbb{R}^{n \times d}$. In this work, we use $\phi(\mathbf{G}) = \mathbf{U}$ where $\mathbf{U}$ is derived by solving the singular value decomposition of $\mathbf{G}$, that is, $\mathbf{G} \approx \mathbf{G}_d \approx \mathbf{U}\mathbf{S}\mathbf{V}^T$ and hence $\mathbf{G}_d$ is the best rank-d approximation of $\mathbf{G}$. The above is only one such possibility of ϕ, as the framework is flexible with many interchangeable components (hence, ϕ can be interchanged with any other function over the sparse adjacency matrix $\mathbf{G}$). In general, $\mathbf{H}^0$ can also be any features, or even random vectors. Hence, they are not required to be dependent on the graph as $\mathbf{H}^0 = \phi(\mathbf{G})$. Furthermore, ℓ can be any function over a graph's adjacency matrix such as the normalized Laplacian or random walk matrix such as $\ell(\mathbf{G}) = \mathbf{Q}^{-\frac{1}{2}}\mathbf{G}\mathbf{Q}^{-\frac{1}{2}}$ where $\mathbf{Q} = \texttt{diag}(\mathbf{G})$ the diagonal node degree matrix of $\mathbf{G}$. Since we also want the model to include the features of the node itself in the propagation, we simply $\mathbf{G} + \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. As an aside, note that for the visualization recommendation problem, we can also set $\mathbf{H}^{(0)}$ to be the meta-feature matrix of the users, attributes, and so on. Now, one simple model is:

$$f(\mathbf{H}^{(k)}, \ell(\mathbf{G})) = \sigma\Big(\ell(\mathbf{G})\mathbf{H}^{(k)}\mathbf{W}^{(k)}\Big) \tag{4}$$

where σ is a non-linear activation function and $\mathbf{W}^{(k)}$ is the weight matrix of the kth layer. An intuitive example is shown in Figure 6. Besides the sum aggregator used implicitly in Eq. 4, we can also

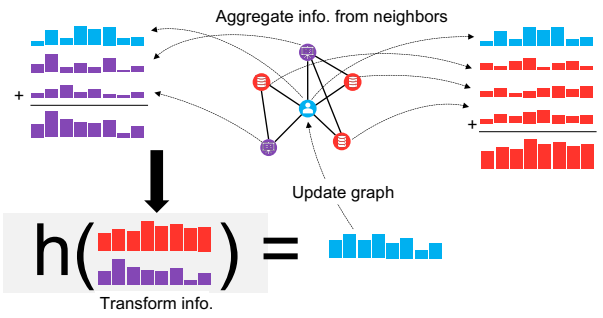

Figure 6: VisGNN aggregates embeddings from the neighboring visual-configurations and data-attributes. The resulting embeddings are then transformed, stored, and updated for use in the next layer.

leverage other relational neighborhood aggregators in our framework, such as the mean aggregator, LSTM aggregator, pooling aggregator, among many others as well. For instance, the mean aggregator would simply be:

$$\mathbf{h}_i^{(k)} = \sigma\big(\mathbf{W}^{(k-1)} \cdot \textsc{mean}(\{\mathbf{h}_i^{(k-1)}\} \cup \{\mathbf{h}_j^{(k-1)},\ \forall j \in N(i)\})\big) \quad (5)$$

More generally,

$$\mathbf{h}_i^{(k)} = \sigma\Big(\mathbf{W}^{(k-1)} \cdot \big[\mathbf{h}_i^{(k-1)}\ \textsc{Aggr}\big(\{\mathbf{h}_j^{(k-1)},\ \forall j \in N(i)\}\big)\big]\Big) \quad (6)$$

where $\textsc{Aggr}(\cdot)$ is any arbitrary aggregator function [28] and $\mathbf{W}^{(k-1)}$ is the learned transformation matrix. This process is repeated for every node in our graph.

The model predicts the probability of an edge (u, v) existing by deriving a score between the representations of node $\mathbf{h}_u$ and $\mathbf{h}_v$ using a function (*e.g.*, an MLP or a dot product):

$$\hat{y}_{uv} = g(\mathbf{h}_u, \mathbf{h}_v) \quad (7)$$

We use a binary cross-entropy loss:

$$\mathcal{L} = -\sum_{uv \in \mathcal{D}} \Big(y_{uv} \log(\hat{y}_{uv}) + (1 - y_{uv}) \log(1 - \hat{y}_{uv})\Big) \quad (8)$$

From the above, we can naturally derive a score for user i on any arbitrary visualization $\mathcal{V}$ by first decomposing it into the visual-configuration t (set of visual design choices) and the attributes used $r_1, \ldots, r_s$. Then, we can easily obtain the probability of each of these components. For instance, for user i and the configuration t, we have y_{it}, and similarly, for attribute r we have y_{ir}. Given all these probabilities, we can combine them to get a probability score for the overall visualization by taking the product. Then, we have

$$\hat{y} = g(\mathbf{h}_i, \mathbf{h}_t) \prod_{j \in \mathcal{A}} g(\mathbf{h}_i, \mathbf{h}_j) \quad (9)$$

where $\hat{y}$ is the final predicted score of the visualization for $\mathcal{V}$ user i. Intuitively, a user-relevant visualization $\mathcal{V}$ is assigned a high score when both the probability of the visual-configuration $g(\mathbf{h}_i, \mathbf{h}_t)$ for user i and the probability of each of the data attributes $j \in \mathcal{A}$, $g(\mathbf{h}_i, \mathbf{h}_j)$ that can be assigned to the visual-configuration are high, that is, $\prod_{j \in \mathcal{A}} g(\mathbf{h}_i, \mathbf{h}_j)$.

Table 1: Summary of the user-level corpus of datasets and visualizations preferred by users from the web (plot.ly).

# Users	17,469
# Datasets	94,419
# Attributes	2,303,033
# Visualizations	32,318
# Vis. Configs	686
mean # attr. per dataset	24.39
mean # attr. per user	51.63
mean # vis. per user	1.85
mean # datasets per user	5.41
Density (A)	<0.0001
Density (C)	<0.0001
Density (D)	<0.0001

In addition, given a user i and a visualization $\mathcal{V} = (\mathbf{X}_{ij}^{(k)}, C_t)$ to score from some dataset, we can leverage the user-specific embeddings learned from our graph neural network to learn another model that outputs a score for a visualization directly. These are leveraged by concatenating the embedding of user i, visual configuration t, along with the embeddings for each attribute $r_1, \ldots, r_s$ used in the visualization. More formally,

$$\psi(\mathcal{V} = \langle \mathbf{X}_{ij}^{(k)}, C_t \rangle) = \big[\mathbf{u}_i\ \mathbf{z}_t\ \mathbf{v}_{r_1}\ \cdots\ \mathbf{v}_{r_s}\big] \quad (10)$$

where $\mathbf{u}_i$ is the embedding of user i, $\mathbf{z}_t$ is the embedding of the visual-configuration C_t, and $\mathbf{v}_{r_1}, \ldots, \mathbf{v}_{r_s}$ are the embeddings of the attributes used in the visualization being scored for user i. For clarity, we use different symbols for each node type, however, each node has a specific index in $\mathbf{H}$, hence, $\mathbf{z}_t$ and $\mathbf{h}_t$ are one in the same.

We leverage the user, visual-configuration, and attribute embeddings as input into a deep multilayer neural network with L fully-connected layers,

$$\psi(\mathcal{V} = \langle \mathbf{X}_{ij}^{(k)}, C_t \rangle) = \big[\mathbf{u}_i\ \mathbf{z}_t\ \mathbf{v}_{r_1}\ \cdots\ \mathbf{v}_{r_s}\big]^\top \quad (11)$$

$$\mathbf{q}_1 = \sigma_1(\mathbf{W}_1 \phi(\mathcal{V}) + \mathbf{b}_1) \quad (12)$$

$$\mathbf{q}_2 = \sigma_2(\mathbf{W}_2 \mathbf{q}_1 + \mathbf{b}_2) \quad (13)$$

$$\vdots$$

$$\mathbf{q}_L = \sigma_L(\mathbf{W}_L \mathbf{q}_{L-1} + \mathbf{b}_L) \quad (14)$$

$$\hat{y} = \sigma(\mathbf{h}^\top \mathbf{q}_L) \quad (15)$$

where $\mathbf{W}_L$, $\mathbf{b}_L$, and σ_L are the weight matrix, bias vector, and activation function for layer L. Further, $\hat{y} = \sigma(\mathbf{h}^\top \mathbf{q}_L)$ (Eq. 15) is the output layer where σ is the output activation function and $\mathbf{h}^\top$ denotes the edge weights of the output function. For the hidden layers, we used ReLU as the activation function. For visualizations that do not use s attributes, then we pad the remaining unused attributes with zeros. This approach allows the multi-layer neural network architecture to be flexible for visualizations with any number of attributes. Hence, $\hat{y}$ is the predicted visualization score for user i.

In addition to visualization recommendation, we can also leverage our GNN-based framework for many other important related visualization tasks including personalized design choice recommendation (*e.g.*, chart-type), personalized attribute recommendation, personalized visualization-configuration recommendation, and even

Table 2: Results for Personalized Visualization Recommendation. Our proposed approach, VisGNN, outperformed all of the baseline models for both the hit ratio (HR) and normalized discounted cumulative gain (NDCG), across all values of k.

	HR@K					NDCG@K				
Model	@1	@2	@3	@4	@5	@1	@2	@3	@4	@5
PopVis	0.105	0.158	0.158	0.158	0.228	0.088	0.132	0.132	0.132	0.156
kNNVis	0.088	0.088	0.105	0.105	0.105	0.028	0.035	0.042	0.048	0.052
eALS	0.111	0.216	0.291	0.349	0.375	0.104	0.172	0.210	0.235	0.245
VisGNN	**0.684**	**0.724**	**0.756**	**0.786**	**0.800**	**0.415**	**0.530**	**0.570**	**0.590**	**0.598**

personalized recommendation of users with similar visual and data preferences or interests (*e.g.*, which may be useful for collaboration purposes, among other applications). Besides visualization recommendation, we also explore using VisGNN for recommending personalized data attributes for users from some user-specific dataset of interest. For a user i, the VisGNN model predicts the probability of a data-attribute j by using the learned representations of the user i denoted as $\mathbf{h}_i$ and the attribute representation $\mathbf{h}_j$ via a function g (such as MLP or a dot product) as follows:

$$\hat{y}_{ij} = g(\mathbf{h}_i, \mathbf{h}_j) \tag{16}$$

Therefore, using the above, we can obtain the probability of each attribute $j \in \mathcal{A}$ in the dataset of interest to user i, that is, $\hat{y}_{ij} = g(\mathbf{h}_i, \mathbf{h}_j), \forall j \in \mathcal{A}$. From the resulting data-attribute probabilities, we obtain a personalized user-relevant ranking of the data-attributes for user i.

4 EXPERIMENTS

We design experiments to investigate the effectiveness of our GNN-based framework for personalized visualization recommendation. In this work, we derived a user-centered dataset where for each user we know their datasets, visualizations, visual-configurations (set of design choices for a visualization), and the subset of attributes used in the visualizations. We started with 2.3 million visualizations from the Plot.ly Community Feed. We group visualizations and datasets by the author user. Each visualization from a user gets decomposed into a visual-configuration (set of design choices) and a set of data attributes used in the visualization (Figure 4). The set of attributes used in the user-generated visualization is typically a small subset of the attributes in the user's dataset. Then, we simply add a node in the graph for a user, the visual-configuration, and a node for every attribute in the dataset (which includes the attributes used in the visualization generated by the user). Nodes are added in the above step only if they do not already exist in the graph. We then add edges that connect a user with the attributes used in the specific visualization they generated, along with an edge connecting the user to the visual-configuration pertaining to the visualization at hand.

Attributes from a dataset that never appeared in a visualization are also included in the graph, since if meta-feature vectors are available for such attributes, then our GNN-based approach can leverage these attributes to learn implicit connections between the attributes that were also never used. For instance, another user that liked or created a visualization using attributes from a different dataset, may be similar to another attribute in some other dataset of interest from a different user through the meta-feature vectors pertaining to these attributes. Hence, connections can be implicitly created through the meta-feature vectors of attributes across different datasets. In addition, even though the attributes are not included in one or more visualizations for a specific user, they are likely to be very useful for generating new visualizations that the user may find useful and insightful. As an example, a dataset of interest to some user may have many attributes, and some of these can be similar to the attributes used in a visualization preferred by that user, and therefore important to them for their underlying task. However, the user may not know about this attribute and its similarity with the attributes of interest (*e.g.*, it could just be overlooked or perhaps the dataset is so large and its infeasible for the user to understand all the attributes in this dataset of interest).

The large corpus of user-preferred visualizations and datasets are summarized in Table 1. In particular, Table 1 reports the number of users, attributes, datasets, visualizations, and visual-configurations extracted from all the visualizations of the users, among many other useful statistics that aid in understanding the large corpus used in this work. The user-level corpus (of datasets and visualizations) consists of a total of 17,469 users; these users created visualizations from 94,419 datasets which included a total of 2,303,033 attributes. The user-centric visualization training corpus has a total of 32,318 relevant visualizations generated by these users with an average of 1.85 relevant visualizations created per user. Each user in the corpus has an average of 5.41 datasets and each dataset has an average of 24.39 attributes. From the 32.3k visualizations, we extracted a total of 686 unique visual-configurations.

For every user and dataset of interest to them, we know the visualizations that they preferred (generated), and therefore can use this ground-truth information to quantitatively evaluate our approach for personalized visualization recommendation. To evaluate the system quantitatively, we randomly select a single relevant visualization generated by the user for a specific dataset of interest to them, and randomly sample 19 negative visualizations that were not of interest to the user.[1] This approach gives us a total of 20 visualizations per user (1 positive + 19 negative) to use for evaluation. Using this held-out set of user visualizations, we evaluate the ability of the proposed approach to recommend positive visualizations to the user (which are visualizations the specific user actually created). In particular, given a user i and a dataset of interest to that user, we use the proposed approach to recommend the top-k visualizations personalized for that specific user and dataset.

[1] Since the space of possible visualizations is tied to the specific user's dataset (and disjoint across different datasets), non-relevant visualizations for a specific dataset and user are sampled from those that can be generated for the underlying dataset.

Table 3: Ablation study results for different variants of our GNN-based framework.

	HR@K					NDCG@K				
Model	@1	@2	@3	@4	@5	@1	@2	@3	@4	@5
VisGNN	**0.684**	0.724	**0.756**	**0.786**	**0.800**	0.415	0.530	0.570	0.590	0.598
VisGNN-LSTM	0.672	**0.727**	0.750	0.763	0.778	**0.530**	**0.633**	**0.653**	**0.661**	**0.667**
VisGNN-MEAN	0.640	0.690	0.726	0.741	0.754	0.399	0.502	0.536	0.549	0.557

Table 4: Results of VisGNN variants that leverage meta-feature embeddings explicitly.

	HR@K				
Model	@1	@2	@3	@4	@5
VisGNN	**0.684**	**0.724**	**0.756**	0.786	0.800
VisGNN-M	0.537	0.660	0.751	**0.825**	**0.873**

To quantitatively evaluate the personalized ranking of visualizations given by the proposed personalized visualization recommendation models, we use rank-based evaluation metrics including Hit Ratio at K (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K) [12]. Intuitively, HR@K quantifies whether the held-out relevant (user-generated) visualization appears in the top-K ranked visualizations or not. Similarly, NDCG@K takes into account the position of the relevant (user generated) visualization in the top-K ranked list of visualizations, by assigning larger scores to visualizations ranked more highly in the list. For both HR@K and NDCG@K, we report $K = 1, \ldots, 5$ unless otherwise mentioned. An effective personalized visualization recommender will assign a larger score to the user-relevant visualizations and smaller scores to the non-relevant visualizations for that specific user.

We used a variety of baseline methods for comparison:

- **PopVis**: PopVis decomposes a visualization into its attribute and visual-configuration, then derives a score for the visualization by taking the product of the number of times the visual-configuration was used in the corpus, along with the frequencies of the attributes used in the visualization of interest.
- **kNNVis**: Given the attributes and visual-configuration of a visualization of interest, we score the visualization by taking the mean score of the visual configurations that are most similar to it, along with the mean score of the top attributes most similar to each of the attributes used in the visualization.
- **eALS**: This baseline is a state-of-the-art MF method typically used for item recommendation [12], which we adapted to our visualization recommendation problem by minimizing squared loss while treating all unobserved user iterations between attributes and visual-configurations as negative examples, which are weighted non-uniformly by the frequency of attributes and visual-configurations.

Personalized Visualization Recommendation Results. Overall, our proposed GNN model for personalized visualization recommendation significantly outperforms the baselines as seen in Table 2. This result holds across both evaluation metrics (HR and NDCG) and across all k. In Table 3, we compare a few variants from our VisGNN framework. For these experiments, we vary the relational aggregation function used in VisGNN while fixing all hyperparameters. This allows us to understand the impact and utility of using other aggregation functions. In particular, we investigate using other relational neighborhood aggregator functions including MEAN and LSTM. For HR@K, we observe that VisGNN generally outperforms the other methods across all hit ratios as shown in In Table 3 (with the exception of HR@K where VisGNN-LSTM performs slightly better). In contrast, we observe that VisGNN-LSTM outperforms the other variants when considering the NDCG ranking evaluation metric. These results demonstrate the utility of graph neural networks for this complex recommendation task.

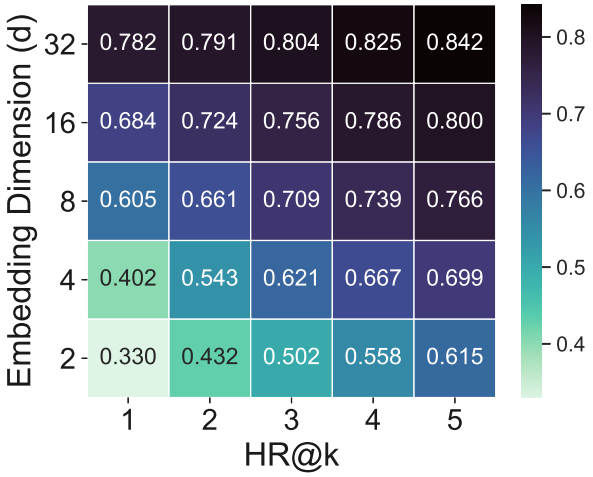

Figure 7: Varying embedding dimension d and HR@K.

VisGNN with Meta-Features. In Table 4, we compare a variant from the VisGNN framework. Notably, we investigate a variant of VisGNN that leverage additional graph information in the form of the meta-feature matrix $\mathbf{M}$. More specifically, we use the meta-feature learning approach proposed in [27] to derive the meta-feature matrix $\mathbf{M}$ that consists of a fixed-length meta-feature vector $\mathbf{m}$ for every attribute across all datasets. Intuitively, the meta-feature vector of an attribute (from an arbitrary dataset) captures the important data characteristics of the attribute in a shared low-dimensional space where attributes from any arbitrary dataset can be compared and leveraged in learning. We map every attribute to a shared k-dimensional meta-feature space that allows our GNN framework to learn from user-level attribute preferences across all the different datasets of the users. Most importantly, the shared meta-feature space is independent of the specific datasets and the meta-features

represent general functions of an arbitrary attribute, independent of the user or dataset that it arises. This approach enables our GNN-based framework to learn from the user-level attribute preferences, despite that those preferences are on entirely different datasets. Now, given $\mathbf{M}$, we derive the following new heterogeneous graph $\mathbf{G}$ as follows:

$$\mathbf{G} = \begin{bmatrix} \blacksquare & \mathbf{A} & \mathbf{C} & \blacksquare \\ \mathbf{A}^\top & \blacksquare & \mathbf{D} & \mathbf{M} \\ \mathbf{C}^\top & \mathbf{D}^\top & \blacksquare & \blacksquare \\ \blacksquare & \mathbf{M}^\top & \blacksquare & \blacksquare \end{bmatrix} \tag{17}$$

Results are provided in Table 4. Overall, we observe that when k is small, the original VisGNN without $\mathbf{M}$ performs best. However, as k becomes larger, then using this additional information during training allows for better user-personalized embeddings to be learned, and therefore improves the user-specific personalized visualization recommendations given by our approach, as shown in Table 4. However, VisGNN-$\mathbf{M}$ still significantly outperforms the baseline methods in Table 2 across all HR@K.

Embedding size. To understand the effect of model performance with respect to the embedding size used in VisGNN, we vary the embedding size d of the VisGNN models from $d \in \{2, 4, 8, 16, 32\}$. We provide results in Figure 7. We observe that performance of the trained VisGNN models generally increases as a function of the embedding size d. More specifically, performance of the VisGNN models generally increases as the embedding size d becomes larger as shown in Figure 7.

Layer size. We also investigate the impact of the layer sizes of VisGNN. In this experiment, the network structure of VisGNN follows a tower pattern where the layer size of each successive layer is halved. In Table 5, we observe a significant improvement in the ranking when using larger layer sizes.

Table 5: Varying the layer sizes used in VisGNN. We vary the layer sizes used in the neural architecture tower structure by a multiple of $\{2, 3, 4\}$.

	HR@K				
Layer Sizes	@1	@2	@3	@4	@5
8-16-32-64-128-256	0.707	0.739	0.769	0.795	0.815
8-28-84-252-756-2268	0.740	0.759	0.772	0.788	0.807
8-32-128-512-2048-8192	**0.794**	**0.804**	**0.817**	**0.837**	**0.851**

Personalized Attribute Recommendation Results. In this section, we investigate VisGNN for personalized attribute recommendation. For these experiments, we randomly hold-out 5% of the nonzero values that correspond to observed attribute preferences. We repeat this 10 times and average the results. Each sample corresponds to a train and test split. The ranking is on the specific dataset where an attribute appears in. For attribute recommendation, we compare to a random baseline that selects an attribute uniformly at random from the test dataset. The probability of correctly recommending the attribute in the test dataset j for user i is $\frac{1}{|\mathbf{X}_{ij}|}$ where $|\mathbf{X}_{ij}|$ denotes the number of attributes (columns) in the data matrix $\mathbf{X}_{ij}$. We also use a kNN baseline for this problem called AttrKNN. This baselines computes the similarity between each of the attributes in the dataset and uses these scores to obtain a ranking of the attributes for the given user. Results are provided in Table 6. Overall, the VisGNN approach outperforms both baselines across all K.

Table 6: Results for Personalized Attribute Recommendation.

	HR@K				
	@1	@2	@3	@4	@5
Random	0.089	0.168	0.262	0.349	0.429
AttrKNN	0.222	0.296	0.296	0.370	0.444
VisGNN	**0.630**	**0.704**	**0.704**	**0.741**	**0.777**

5 CONCLUSION

This work proposed VisGNN: a graph neural network framework for the problem of personalized visualization recommendation. To the best of our knowledge, this is the first work to develop and leverage GNNs for this problem. We developed a GNN-based framework that first represents the large corpus of datasets and visualizations from users as a large heterogeneous graph. Our GNN framework decomposes a visualization into its data and visual components, and then jointly models each of them as a large graph to obtain embeddings of the users, attributes (across all datasets in the corpus), and visual-configurations. From these user-specific embeddings of the attributes and visual-configurations, we can then predict the probability of any visualization arising from a specific user. Finally, the experiments demonstrated the effectiveness of using graph neural networks for automatic and personalized recommendation of visualizations to specific users based on their data and visual (design choice) preferences.

REFERENCES

[1] Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 1957–1967. https://doi.org/10.18653/v1/D17-1209

[2] Stephen M Casner. 1991. Task-Analytic Approach to the Automated Design of Graphic Presentations. *ACM Transactions on Graphics (ToG)* 10, 2 (1991), 111–151.

[3] Zhe Cui, Sriram Karthik Badam, M Adil Yalçin, and Niklas Elmqvist. 2019. Datasite: Proactive Visual Data Exploration With Computation of Insight-Based Recommendations. *Information Visualization* 18, 2 (2019), 251–267.

[4] Tuan Nhon Dang and Leland Wilkinson. 2014. ScagExplorer: Exploring Scatterplots by Their Scagnostics. In *2014 IEEE Pacific visualization symposium*. IEEE, 73–80.

[5] Çağatay Demiralp, Peter J Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. 2017. Foresight: Recommending Visual Insights. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 10.

[6] Mark Derthick, John Kolojejchick, and Steven F Roth. 1997. An interactive visualization environment for data exploration. In *KDD*. 2–9.

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

[8] Stef van den Elzen and Jarke J. van Wijk. 2013. Small Multiples, Large Singles: A New Approach for Visual Data Exploration. In *Computer Graphics Forum*, Vol. 32. 191–200.

[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[10] Steven Feiner. 1985. APEX: An Experiment in the Automated Creation of Pictorial Explanations. *IEEE Computer Graphics and Applications* 5, 11 (1985), 29–37.

[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for

recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.
[12] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 549–558.
[13] Kevin Hu, Diana Orghian, and César Hidalgo. 2018. Dive: A mixed-initiative system supporting integrated data exploration workflows. In *Workshop on Human-In-the-Loop Data Anal.* 1–7.
[14] Alicia Key, Bill Howe, Daniel Perry, and Cecilia Aragon. 2012. VizDeck: Self-Organizing Dashboards for Visual Analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 681–684.
[15] Doris Jung-Lin Lee. 2020. *Insight Machines: The Past, Present, and Future of Visualization Recommendation.*
[16] Doris Jung-Lin Lee, Himel Dev, Huizi Hu, Hazem Elmeleegy, and Aditya Parameswaran. 2019. Avoiding Drill-Down Fallacies With VisPilot: Assisted Exploration of Data Subsets. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 186–196.
[17] Halden Lin, Dominik Moritz, and Jeffrey Heer. 2020. Dziban: Balancing Agency & Automation in Visualization Design via Anchored Recommendations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.
[18] Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. 2018. DeepEye: towards automatic data visualization. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 101–112.
[19] Jock Mackinlay. 1986. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.* 5, 2 (1986), 110–141.
[20] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show Me: Automatic presentation for visual analysis. *TVCG* 13, 6 (2007), 1137–1144.
[21] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show Me: Automatic Presentation for Visual Analysis. *IEEE transactions on visualization and computer graphics* 13, 6 (2007), 1137–1144.
[22] Dominik Moritz, Chenglong Wang, Greg L Nelson, Halden Lin, Adam M Smith, Bill Howe, and Jeffrey Heer. 2018. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 438–448.
[23] Belgin Mutlu, Eduardo Veas, and Christoph Trattner. 2016. Vizrec: Recommending personalized visualizations. *ACM Transactions on Interactive Intelligent Systems (TIIS)* 6, 4 (2016), 1–39.
[24] Daniel B Perry, Bill Howe, Alicia MF Key, and Cecilia Aragon. 2013. VizDeck: Streamlining exploratory visual analytics of scientific data. (2013).
[25] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. 2018. Learning Human-Object Interactions by Graph Parsing Neural Networks. arXiv:1808.07962 [cs.CV]
[26] Xin Qian, Ryan A Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Nesreen K Ahmed. 2021. Personalized Visualization Recommendation. *arXiv:2102.06343* (2021).
[27] Xin Qian, Ryan A Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Joel Chan. 2021. Learning to Recommend Visualizations from Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1359–1369.
[28] Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. 2017. Deep feature learning for graphs. *arXiv preprint arXiv:1704.08829* (2017).
[29] Steven F Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. 1994. Interactive graphic design using automatic presentation knowledge. In *CHI*. 112–117.
[30] Jinwook Seo and Ben Shneiderman. 2005. A Rank-by-Feature Framework for Interactive Exploration of Multidimensional Data. *Information visualization* 4, 2 (2005), 96–113.
[31] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless Data Exploration With zenvisage: An Expressive and Interactive Visual Analytics System. *arXiv preprint arXiv:1604.03583* (2016).
[32] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *TVCG* 8, 1 (2002), 52–65.
[33] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. arXiv:1706.02263 [stat.ML]
[34] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. 2017. Towards visualization recommendation systems. *SIGMOD* 45, 4 (2017), 34–39.
[35] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 8. NIH Public Access, 2182.
[36] Martin Voigt, Stefan Pietschmann, and Klaus Meißner. 2013. A semantics-based, end-user-centered information visualization process for semantic web data. In *Semantic models for adaptive interactive systems*. Springer, 83–107.
[37] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.
[38] Leland Wilkinson and Graham Wills. 2008. Scagnostics Distributions. *Journal of Computational and Graphical Statistics* 17, 2 (2008), 473–491.
[39] Graham Wills and Leland Wilkinson. 2010. Autovis: automatic visualization. *Information Visualization* 9, 1 (2010), 47–69.
[40] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Towards a General-Purpose Query Language for Visualization Recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM, 4.
[41] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE transactions on visualization and computer graphics* 22, 1 (2016), 649–658.
[42] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2648–2659.
[43] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? arXiv:1810.00826 [cs.LG]
[45] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.
[46] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.