Visualization Recommender

Neha Pradip Jagtap[1] and Prof. Dr. Zeshan Afzal[2]

[1] njagtap@rhrk.uni-kl.de
[2] muhammad-zeshan.afzal@dfki.de

Abstract. Extracting valuable insights from a dataset is a crucial part of any domain study. For this purpose visualization plays an important role. Creating effective visualizations with expressive visualization tools often requires users to manually specify views, which can be a daunting task for those with limited statistical knowledge and coding skills. As a solution, visualization recommender systems have emerged, with the goal of lowering the barrier to exploratory data analysis by automating the process of identifying and recommending appropriate visualizations. Our work presents a novel approach to visualization recommendation, leveraging an end-to-end trainable model to automatically generate visualization recommendations. By automating the visualization process, this approach eliminates the need for manual specification and can be particularly beneficial for users with limited time or resources. The proposed model represents a promising step towards more accessible and efficient exploratory data analysis. In the past we have systems which tells us how we can visualize data in better ways, which charts can be made, what styles to use etc. But the important thing which comes even before the actual visualization is, what data to visualize. Here we are trying to address this question as well.

Keywords: Visualization Recommender Systems, Feature Extraction, Deep Learning

1 Introduction

Visualization recommender systems can provide recommendations in two forms: what data to visualize or how to visualize the selected data. The former is known as data query recommendation and has various approaches. Recent research has focused on optimizing statistical functions for data query recommendation. It is important to note that data query specification is a crucial aspect of visualization and it is a separate task from design choice recommendation.
The Visualization Recommender system utilizes machine learning techniques to recommend visualizations based on a vast collection of datasets and associated visualizations. By automating the process of suggesting design choices that optimize effectiveness, the system aims to minimize the need for manual creation of KPI´s for visualizations, resulting in significant cost reduction. Helps for finding KPIs in the dataset.
Plotly datasets with multiple visualizations are used for training. The learning task at hand is to optimize this approach(Training and inference pipeline) that uses features of datasets to predict these column and design choices. The system proposed is a machine learning-based method for recommending visualizations, which utilizes a vast dataset and associated visualizations. The objective of visualization recommendation is to develop models that learn to make the most effective design choice. To achieve this, the machine learning model is trained and validated using distinct dataset-visualization pairs sourced from Plotly.
The remainder of this paper is structured as follows. Section 2 provides a survey of the existing literature. In Section 3, we outline our approach to the problem, which includes details on the dataset used, feature extraction, and the development of our model. We then move on to discuss the results and experiments conducted in Section 4. Finally, in Section 5, we present our conclusions, highlighting the potential applications and improvements of our proposed method, as well as identifying emerging research directions.

2 Related Work

Visualization recommender systems are designed to assist users in selecting what data to visualize and how to visualize it. They can help users make sense of complex data by suggesting appropriate visualizations based on the characteristics of the data and the user's goals.
One type of visualization recommender system is a data query recommender, which suggests which data to visualize based on the user's query. These systems use statistical functions to optimize the query specification for effective visualization. Another type of visualization recommender system focuses on design choice recommendation, which suggests how to visualize the selected data.

Rule-based Visualization Recommender Systems: These systems typically use guidelines specified by experts, such as Bertin[1] or Cleveland and McGill[2], to score visualizations based on effectiveness criteria.Popular visualization recommender systems include SAGE[8] and Show Me[6], which use data, encodings, and task types to improve effectiveness criteria.
Recently, systems such as Voyager[9] have combined rule-based approaches with visualization recommendations derived from additional information to provide more effective visualization recommendations.

Machine learning-based visualization recommender systems: Offer an end-to-end learning approach that eliminates the need for hand-engineered features.
Data2Vis [3] presents a novel approach to map the correlation between JSON encoded data rows and successful visualization specifications. The visualization task is treated as a neural machine translation problem, where a sequence-to-sequence deep neural network learns the relationship between data rows and visual specifications. The model learns the syntax of Vega-Lite grammar, as the input is a JSON object. The authors claim that the method can be adapted to work with other visualization modes given sufficient data. However, Data2Vis only considers single rows as input and does not take into account inter-row relations.
VizML[4] employs the concept of pairwise correlated column features, using a feature vector of aggregated column statistics as input to a simple neural network to predict a partial visual specification. Unlike other methods that recommend a combination of both machine learning and heuristics, VizML uses a purely visual encoding-based approach to visualization recommendation. The predicted encoding includes the type of visualization, limited to bar, line, and scatter plots, and the axis on which a column is realized. However, VizML's work is limited as it can only predict partial view specifications of a limited set of charts
DeepEye[5] utilizes a decision tree classifier to learn statistical features from expert-annotated data, which are used to filter out bad visualizations and rank the remaining ones using a heuristic-based approach. The trained model was then used to filter out bad visualizations from all possible visualizations for a given dataset. The system also employs heuristics and a ranking algorithm to generate the final recommendation. To achieve this, 14 pairwise correlated column statistical features, along with the features from individual columns, were calculated using a mini-grammar that supports column transformations and feature aggregation. These features were visualized as a bar, line, scatter, or pie chart. Thus, DeepEye's recommendation system uses a hybrid approach that combines machine learning with a heuristic-based ranking system to guide the recommendation.
Draco[7] presents visualization recommendation as a constraint optimization problem. By assigning costs to constraint violations, Draco learns the weights for soft constraints that can be violated. The RankSVM algorithm then chooses between competing heuristics based on pairs of visualizations, with the heuristics themselves being based on user conditions and contextual information, such as column importance. Draco generates visualizations by solving a combinatorial constraint optimization problem and expresses the specifications using Vega-Lite grammar. However, the authors acknowledge that Draco only considers a subset of Vega-Lite and therefore lacks expressivity and many features.

3 Methodology

In order to predict the possible visualizations which will provide key information from the dataset, We have approached the problem in the following steps as shown in figure 1.

1. Data Collection: In this study, we performed data collection by acquiring various types of table datasets from Plotly. These datasets were obtained from different fields and had varying numbers of columns. Additionally, the datasets contained data of different data types.
2. Data Cleaning: we performed data cleaning by conducting a brief survey to identify possible problems in the raw data. We addressed these problems based on our requirements. The basic data cleaning process is summarized in table 2. After preparing the cleaned data, we defined the Key Performance Indicators (KPIs) for each dataset. Then, we created visualizations based on the KPIs to analyze and present the data effectively.
3. Feature Extraction: We map each dataset to 852 features, mapped from 77 single column features, 38 semantic-column features and 622 aggregate column features using 16 aggregation functions. After feature extraction an integrity check was performed to ensure the correctness of these features.
4. Model Training: We trained our neural network on the extracted features and predicted the possible visualizations.

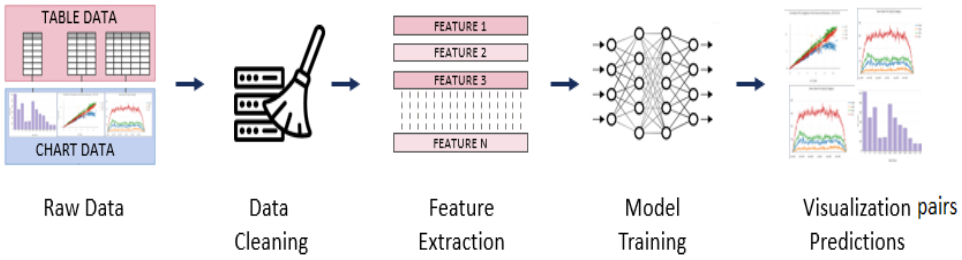

Fig. 1: Workflow of the Visualization Recommender

3.1 Datasets

We utilized publicly available datasets from Plotly as the dataset of interest. Prior to analysis, each dataset was subject to a thorough data cleaning process that involved the removal of missing, duplicate, corrupt, and erroneous values as shown in table2. Using predefined Key Performance Indicators (KPIs) for each dataset, we generated visualizations such as bar , line, and pie charts to present the data in an intuitive manner. To facilitate the selection of appropriate visualization techniques for each dataset, we also created an adjacency matrix that establishes relationships between pairs of columns in the dataset and determines the feasibility of creating a chart with that specific pair.

COMMON ISSUES	IMPLEMENTED SOLUTIONS
Duplicate Data	Removal of Duplicate Date
Missing values	Removal of such Rows
Irrelevant data	Irrelevant to our problem
Unwanted Outliers	Irrelevant to our problem
Single Value Columns	Removed such rows
Structural Errors (e.g caps /non caps)	Catered in dataset creation

Fig. 2: Data cleaning tasks handled

3.2 Feature Extraction

We have extracted 3 types of features from the data namely, single column features, aggregate column features and the semantic features, see fig 4. Overall 77 single column features including quantitative features, string features etc are extracted for each column of the dataset. Then we calculate all the unique combinations of the pairs in the dataset and extract their pairwise aggregate features(622). For extracting the semantics for the columns we refereed the work from [10]. Since we have different special cases we modified the semantics as per the requirement and came up with 38 such unique semantic features which can represent a column in dataset uniquely as close as possible.

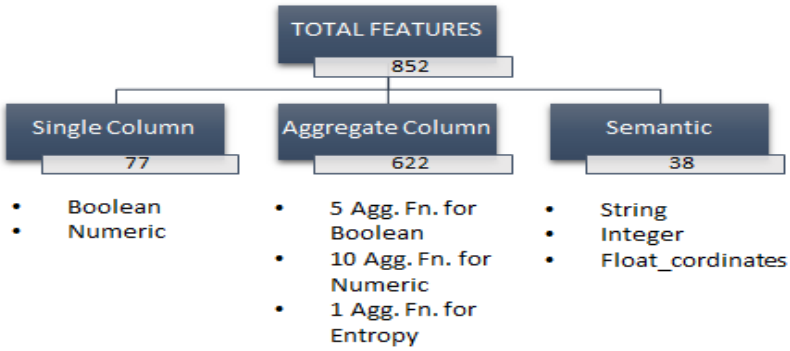

Fig. 3: Features Extraction Explained

The four categories of single column features are as follows:

1. The dimension feature which encodes the number of rows in a column.
2. The types feature which encodes the type of data present in a column, such as categorical, temporal or quantitative.
3. The values feature which captures the statistical properties of columns.

4. The name feature which contains the column names. This categorization is crucial for accurate feature computation.

In total, there are 16 aggregation operations that are used to transform the single column features into a scalar value for each pair of columns. For quantitative column features, aggregation functions such as mean, variance, and standard deviation are utilized. Conversely, aggregation operations like percent, has, only one, and all are only applied to categorical features. As a result, we obtain 622 features for each pair..

The 38 semantic features include the string semantics, integer semantics, float and cordinate semantics. So for each column we have 77 single column features and 38 semantic features. Along with them 622 aggregate features for each possinle pair of columns which may be then used for visualization purpose.

3.3 Training Neural Network

Our model consists of a simple neural network which takes input as the number of features for each pair, which is 852. We gradually decrease the number of output neurons from 512 to 256 till a single output scalar value.. Our primary model is a fully-connected neural network (NN) with 4 hidden layers. The first hidden layer has 512 units, the second has 256 units, the third has 128 units, and the fourth has 64 units with LeakyReLU activation functions and implemented using PyTorch. The neural network was trained with the Adam optimizer. The learning rate was initialized at 3×10^{-4}. For comparison, we chose these classifiers with default parameters: K-nearest neighbors Classifier, Gaussian Process Classifier, Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier and Neural Net(Multilayer perceptron).

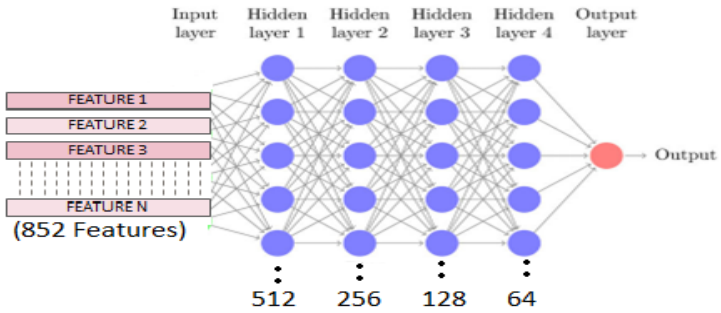

Fig. 4: Neural Network Architectute

4 Experiments

We have very good results for our model. Our model was able to perform very well with 95.25 percentage validation accuracy with loss of 0.1036 for 100 epochs, training accuracy is 94.29 percentage with loss of 0.0796 and test accuracy of 95.46 percentage. We also have tested the results on the following classifiers as well. I have listed all the results bellow. We have also used evaluation metrics like f1 score, recall and precision to verify the results. We have observed that even though we had very low positives in our dataset, the models were able to predict them really well. We tested on multiple different datasets and observed similar results with minor change in accuracies.

Classifier	Test Accuracy
Random Forest	94.38%
Nearest Neighbors	98.61%
Decision Tree	98.74%
AdaBoost	98.11%
Neural Net	98.26%
Gaussian Process	98.42%

Fig. 5: Comparison of accuracies

5 Conclusion

The problem of recommending visualizations has been approached in different ways, including rule-based constraint optimization and visualization grammars, also combinations of these methods with machine learning. While recent research has focused on purely machine learning-based systems.
In this work, we build on previous research by formulating visualization recommendation as a purely machine learning problem using a neural network model. Our model demonstrates effectiveness in predicting possible visualizations and we provide a mechanism for preliminary visualization. We have successfully increased the predictive power of the model by combining the semantic features with the quantitative features which improved the performance.
However, the system is currently limited to preliminary visualizations and does not support interactive visualizations. Also currently we are considered two columns for visualizations which can further be extended for multiple column visualizations. Future work also can include the more complicated chart types. Additionally, it has been trained using the Plotly corpus, which may introduce bias towards data sourced from Plotly. Further training with datasets from different sources can mitigate this bias.

References

1. Jacques Bertin. *Semiology of graphics*. University of Wisconsin press, 1983.
2. William S Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984.
3. Victor Dibia and Çağatay Demiralp. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications*, 39(5):33–46, 2019.
4. Kevin Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César Hidalgo. VizML: A Machine Learning Approach to Visualization Recommendation. In *Proceedings of the 2019 Conference on Human Factors in Computing Systems (CHI)*. ACM, 2019.
5. Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. Deepeye: Towards automatic data visualization. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 101–112. IEEE, 2018.
6. Jock Mackinlay, Pat Hanrahan, and Chris Stolte. Show me: Automatic presentation for visual analysis. *IEEE transactions on visualization and computer graphics*, 13(6):1137–1144, 2007.
7. Dominik Moritz, Chenglong Wang, Greg L Nelson, Halden Lin, Adam M Smith, Bill Howe, and Jeffrey Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics*, 25(1):438–448, 2018.
8. Steven F Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. Interactive graphic design using automatic presentation knowledge. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 112–117, 1994.
9. Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics*, 22(1):649–658, 2015.
10. Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. Sato: Contextual semantic type detection in tables. *Proc. VLDB Endow.*, 13(12):1835–1848, 2020.