

# Automating Data Visualization through Recommendation

by

Kevin Zeng Hu

S.M., Massachusetts Institute of Technology (2015)

S.B., Massachusetts Institute of Technology (2013)

Submitted to the Program in Media Arts and Sciences,  
School of Architecture and Planning,  
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

© Massachusetts Institute of Technology 2019. All rights reserved.

**Signature redacted**

Author .....

Program in Media Arts and Sciences

May 13, 2019

**Signature redacted**

Certified by .....

César Hidalgo

Associate Professor of Media Arts and Sciences

MIT Media Lab

Thesis Supervisor

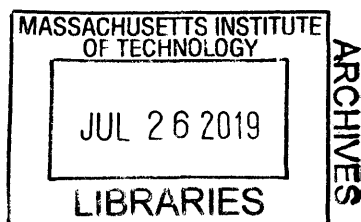
Accepted by .....

**Signature redacted**

Todd Machover

Academic Head

Program in Media Arts and Sciences



# Automating Data Visualization through Recommendation

by

Kevin Zeng Hu

Submitted to the Program in Media Arts and Sciences,  
School of Architecture and Planning,  
on May 13, 2019, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy in Media Arts and Sciences

## Abstract

Demand for data visualization has exploded in recent years with the increasing availability and use of data across domains. Traditional visualization techniques require users to manually specify visual encodings of data through code or clicks. While manual specification is necessary to create bespoke visualizations, it renders visualization inaccessible to those without technical backgrounds. As a result, visualization recommender systems, which automatically generate results for users to search and select, have gained popularity. Here, I present systems, methods, and data repositories to contextualize and improve visualization recommender systems.

The first contribution is **DIVE**, a publicly available and open source system that combines rule-based recommender systems with manual specification. DIVE integrates state-of-the-art data model inference, visualization, statistical analysis, and storytelling capabilities into a unified workflow. In a controlled experiment, we show that DIVE significantly improves task performance among a group of 67 professional data scientists. Over 15K users have uploaded 7.5K datasets to DIVE since its release.

In response to the limitations of rule-based recommender systems, **VizML** is a machine learning-based method for visualization recommendation. VizML uses neural networks trained on a large corpus of datasetvisualization pairs to predict visualization design choices, such as visualization type and axis encoding, with an accuracy of over 85%, exceeding that of base rates and baseline models. Benchmarking with a crowdsourced test set, we show that our model achieves human-level performance when predicting consensus visualization type.

To support learned visualization systems, **VizNet** is a large-scale visualization learning and benchmarking repository consisting of over 31M real-world datasets. To demonstrate VizNet’s utility as a platform for conducting crowdsourced experiments

with ecologically valid data, we replicate a prior perceptual effectiveness study, and demonstrate how a metric of visualization effectiveness can be learned from experimental results. Our results suggest a promising method for efficiently crowdsourcing the annotations necessary to train and evaluate machine learning-based visualization recommendation at scale.

Enabled by the availability of real-world data, **Sherlock** is a deep learning approach to semantic type detection. We train Sherlock on 686K data columns retrieved from the VizNet corpus by matching 78 semantic types from DBpedia to column headers. We characterize each matched column with 1,588 features describing the statistical properties, character distributions, word embeddings, and paragraph vectors of column values. A multi-input neural network achieves a support-weighted F1 score of 0.89, exceeding that of a decision tree baseline, dictionary and regular expression benchmarks, and the consensus of crowdsourced annotations.

I conclude by discussing three opportunities for future research. The first describes design considerations for mixed-initiative interactions in AI-infused visualization systems such as DIVE. The second reviews recent work on statistical validity of insights derived from visualization recommenders, which is an especially important consideration with learned systems such as VizML. Lastly, I assess the benefits of learning visualization design from non-experts then present experimental evidence towards measuring the gaps between expert and non-expert judgment.

Thesis Supervisor: César Hidalgo

Title: Associate Professor of Media Arts and Sciences, MIT Media Lab

# Automating Data Visualization through Recommendation

by

Kevin Zeng Hu

The following people served as readers for this thesis:

Thesis Reader, **Signature redacted** .....  
Tim Kraska  
Associate Professor of Electrical Engineering and Computer Science  
MIT Computer Science and Artificial Intelligence Laboratory

Thesis Reader, **Signature redacted** .....  
Arvind Satyanarayan  
Assistant Professor of Electrical Engineering and Computer Science  
MIT Computer Science and Artificial Intelligence Laboratory

## Acknowledgments

My time at the MIT Media Lab began in January 2012 when I started as an undergraduate intern under Shahar Ronen. Shahar was a student in the Collective Learning (née Macro Connections) group led by the charismatic and brilliant Professor César Hidalgo. Little did I know at the time that César would become the greatest teacher of my life. He taught me how to conduct research, showed me how to write, and supported me through challenging times.

I have been extremely indebted to many teachers throughout and beyond MIT. The members of my doctoral thesis committee, Tim Kraska and Arvind Satyanarayan, together with Çagatay Demiralp, encouraged me to pursue challenging questions and showed me how to communicate research to a broader visualization and machine learning community.

Although I cannot repay the debt I owe to my teachers, I tried to pay it forward by mentoring undergraduate interns over the years. Helping them learn and grow and start amazing careers has been a privilege.

I am also grateful for a diverse group of peers and collaborators. Throughout its multiple generations of students, the Collective Learning group was my family at the lab. Thank you Cristian Jara Figueroa, Tarik Roukny Ornia, Flavio Pinheiro, Diana Orghian, Sanjay Guruprasad, Deepak Jagdish, Mary Kaltenberg, and many others. Beyond the group, it has been a privilege to work with talented peers at the Media Lab and MIT more broadly, like Michiel Bakker, Madelon Hulsebos, Owais Khan, Travis Rich, and Alexis Hope.

This journey would have been much more challenging, and definitely less enjoyable, without good company along the way. Though I'm lucky to call my peers some of my best friends, my research would not have been possible without friends from all

walks of life. There are three groups of friends I'd like to thank in particular. Suite C, we've known each other for almost a decade and I look forward to the decades to come. Funfetti, you were my first friends at the Media Lab and made it into a home. Revengers, I couldn't ask for better friends to travel the world with.

Above all, I am thankful for my family. But because of them, I'm not the first Dr. Hu. That honor goes to my father, Haoran Hu, an man of extraordinary vision and discipline. He taught me that it is possible to achieve one's dreams – but only through hard work. Nor am I the second Dr. Hu. Caroline Hu is one of few people that is world-class at three things: science, art, and being a great Siss. While our height difference has inverted since childhood, I will always look up to her.

Most of all, as I write this on Mother's Day 2019, I am grateful to my late mother, Ping Zeng Hu. Mama Hu taught me all that is worth knowing in this life: how to learn, how to love, and how to be happy. She deserves all the credit for Dr. Hu #3.

# Contents

<b>Abstract</b>	<b>2</b>
<b>1 Introduction</b>	<b>19</b>
1.1 Problem . . . . .	20
1.2 Approach . . . . .	21
1.3 Outline and Contributions . . . . .	23
1.4 Prior Publications and Collaboration . . . . .	26
<b>2 Related Work</b>	<b>28</b>
2.1 Visualization Systems . . . . .	29
2.1.1 Dataset-Specific Systems . . . . .	29
2.1.2 Dataset-Agnostic Systems . . . . .	31
2.2 Visualization Recommenders . . . . .	33
2.2.1 Data Query Recommenders . . . . .	34
2.2.2 Rule-based Recommenders . . . . .	34
2.2.3 Machine Learning-based Recommenders . . . . .	35
2.3 Data Collection From and For Visualization Research . . . . .	37
2.3.1 Graphical Perception . . . . .	38
2.3.2 Data Collection for Visualization Research . . . . .	39
2.3.3 Machine Learning Corpora . . . . .	39
2.4 Semantic Type Detection . . . . .	39

2.4.1	Matching-based . . . . .	40
2.4.2	Data-driven . . . . .	41
2.5	Summary . . . . .	42
<b>3</b>	<b>DIVE</b>	<b>43</b>
3.1	Usage Scenario . . . . .	45
3.2	Design Considerations . . . . .	47
3.2.1	Discretize Workflow into Tasks Grouped by Ordered Modes . . . . .	47
3.2.2	Hierarchically Distinguish Between Navigation, Configuration, and Results . . . . .	48
3.2.3	Organize Input Components into Tightly-Coupled Hierarchies . . . . .	49
3.2.4	Combine Populated Defaults with Incremental Selection . . . . .	49
3.2.5	Distinguish Recommendation Types . . . . .	49
3.3	System Description . . . . .	51
3.3.1	Datasets: Upload and Inspect . . . . .	51
3.3.2	Visualize: Explore and Drill-down . . . . .	53
3.3.3	Analysis: Aggregation, Correlation, Comparison, and Regression . . . . .	54
3.3.4	Stories: Compose and Share . . . . .	55
3.4	Architecture and Implementation . . . . .	55
3.5	Evaluation: DIVE Versus Excel . . . . .	56
3.5.1	Participant Description . . . . .	57
3.5.2	Experimental Procedure . . . . .	57
3.5.3	Experiment Results . . . . .	61
3.5.4	Qualitative Participant Feedback . . . . .	64



3.5.5	Experimental Results Summary . . . . .	64
3.6	Summary and Future Work . . . . .	65
<b>4</b>	<b>VizML</b>	<b>67</b>
4.1	Problem Formulation . . . . .	69
4.1.1	Modeling Design Choice Recommendation . . . . .	71
4.2	Data . . . . .	72
4.2.1	Collection and Cleaning . . . . .	73
4.2.2	Data Description . . . . .	75
4.2.3	Feature Extraction . . . . .	77
4.2.4	Design Choice Extraction . . . . .	79
4.3	Methods . . . . .	79
4.3.1	Feature Processing . . . . .	81
4.3.2	Prediction Tasks . . . . .	81
4.3.3	Neural Network and Baseline Models . . . . .	83
4.4	Evaluating Performance . . . . .	84
4.4.1	Interpreting Feature Importances . . . . .	87
4.5	Benchmarking with Crowdsourced Effectiveness . . . . .	90
4.5.1	Modeling and Measuring Effectiveness . . . . .	90
4.5.2	Data Preparation . . . . .	92
4.6	Crowdsourced Evaluation Procedure . . . . .	93
4.6.1	Benchmarking Procedure . . . . .	95
4.6.2	Benchmarking Results . . . . .	96
4.7	Discussion . . . . .	99

4.8	Future Work . . . . .	100
<b>5</b>	<b>VizNet</b>	<b>102</b>
5.1	Data . . . . .	104
5.1.1	Corpora . . . . .	104
5.1.2	Characterization . . . . .	105
5.2	Experiment Design . . . . .	107
5.2.1	Replication of Kim and Heer (2018) . . . . .	108
5.2.2	Datasets . . . . .	108
5.2.3	Visual Encodings . . . . .	109
5.2.4	Tasks . . . . .	110
5.2.5	Procedure . . . . .	110
5.2.6	Participants . . . . .	111
5.3	Results . . . . .	112
5.3.1	Comparing Subject Performance . . . . .	112
5.3.2	Extending with an Outlier Detection Task . . . . .	113
5.3.3	Learning a Model to Predict Effectiveness . . . . .	115
5.3.4	Limitations . . . . .	116
5.4	Discussion . . . . .	118
5.5	Summary and Future Work . . . . .	120
<b>6</b>	<b>Sherlock</b>	<b>123</b>
6.1	Related Work . . . . .	126
6.2	Data . . . . .	129

6.2.1	Data Collection . . . . .	129
6.2.2	Feature Extraction . . . . .	130
6.2.3	Filtering and Preprocessing . . . . .	132
6.3	Methods . . . . .	133
6.3.1	Sherlock: A Multi-input Neural Network . . . . .	133
6.3.2	Benchmarks . . . . .	135
6.3.3	Training and Evaluation . . . . .	138
6.4	Results . . . . .	138
6.4.1	Benchmark Results . . . . .	138
6.4.2	Performance for Individual Types . . . . .	140
6.4.3	Contribution by Feature Category . . . . .	141
6.4.4	Feature Importances . . . . .	141
6.4.5	Rejection Curves . . . . .	143
6.5	Discussion . . . . .	144
6.6	Summary and Future Work . . . . .	144
<b>7</b>	<b>Conclusion</b>	<b>148</b>
7.1	Review of Contributions . . . . .	148
7.2	Future Work . . . . .	150
7.2.1	Mixed-Initiative Interactions . . . . .	151
7.2.2	Statistical Validity of Visual Analysis . . . . .	156
7.2.3	Experts versus Non-experts . . . . .	157
7.3	Closing Remarks . . . . .	160

# List of Figures

1-1	Example specification for a 2D scatterplot visualizing two columns of the classic automobile dataset [142]. . . . .	22
1-2	Creating a visualization in Vega-Lite and Tableau. . . . .	22
1-3	Graphical Outline of the Thesis . . . . .	24
3-1	Four stages of a data exploration workflow in DIVE: (A) Datasets, (B) Visualize, (C) Analysis, and (D) Stories. . . . .	43
3-2	The DIVE user interface. The left navigation bar labeled (A) is used to navigate between modes and modify project properties. The top navigation bar labeled (B) marks the current user state (project, dataset, and mode) and lets users switch projects or datasets. The main pane labeled (C) displays the main results of the mode. The right selection menu (D) lets users change mode-specific parameters (D1) or selecting fields (D2). All other modes in DIVE follow this hierarchical four-section layout, though some modes do not include a selection menu (D). . . . .	45
3-3	Map of the four user stages (rows) and associated tasks (cells) in DIVE. Tasks are connected by an arrow if they are connected by an action. A complete, linear use case of DIVE would begin on the top-left [1. Datasets > Upload] task and progress downwards through modes, towards the bottom-center [4. Stories > Story] task. . . . .	47
3-4	Recommended visualizations in the Explore mode, as fields incrementally selected. (A) shows the default view when a user first navigates to Explore, while (B), show recommended visualizations if three fields ( <b>position</b> , <b>year</b> , <b>salary</b> ) are selected. The sections marked (B1), (B2), (B3), and (B4) contain exact, subset, individual, and expanded matches, respectively. . . . .	50
3-5	Inspect mode table view showing the following inferred field properties: (A) name, (B) ID or univariate descriptive visualization, (C) statistical properties, (D) field color and selectors for changing the color or field type, and (E) data sample. . . . .	52

3-6	Three-stage visualization recommendation system. Given a user data model and user field selection, the system first enumerate visualization specifications, then materializes the specifications and filters based on effectiveness and expressiveness criteria, and finally scores the visualizations. . . . .	53
3-7	DIVE system diagram. . . . .	56
4-1	A diagram of the data processing and analysis flow in VizML, starting from (1) the original Plotly Community Feed API endpoints, proceeding to (2) the deduplicated dataset-visualization pairs, (3a) features describing each individual column, pair of columns, and dataset, (3b) the design choices extracted from visualizations, (4) task-specific models trained on these features, and (5) potential recommended design choices. . . . .	67
4-2	Example specification for a 2D scatterplot visualizing two columns of the Cars dataset. . . . .	69
4-3	Creating visualizations is a process of making design choices, which can be recommended by a system or specified by an analyst. . . . .	70
4-4	Basic setup of learning models to recommend design choices with a corpus of datasets and corresponding design choices. . . . .	72
4-5	Creating a scatterplot in the Plotly schema using the Plotly Chart Builder and the Plotly Python Library. . . . .	73
4-6	Screenshot of the Plotly Community Feed [132]. . . . .	74
4-7	Distribution of plots per user, visualized on a log-linear scale. . . . .	75
4-8	Distribution of dataset dimensions in the Plotly corpus. . . . .	76
4-9	Extracting features from the Automobile MPG dataset. [142] . . . . .	77
4-10	Extracting design choices from a dual-axis scatterplot visualizing three columns of the MPG dataset. . . . .	77

4-11	Marginal contribution to neural network accuracy by feature set, for each task. Baseline accuracies are shown as solid and dashed lines for naive Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR), and random forest (RF). <b>HSA</b> = Has Shared Axis, <b>ISA</b> = Is Shared X-axis or Y-Axis and <b>XY</b> = Is on X-axis or Y-axis. . . . .	85
4-12	Experiment flow. The original user-generated visualizations are highlighted in blue, while we generated the visualizations of the remaining types. After crowdsourced evaluation, we have a set of votes for the best visualization type of that dataset. We calculate confidence intervals for model scores through bootstrapping. . . . .	94
4-13	Distribution of Gini coefficients for the visualization type prediction tasks. Higher Gini indicates stronger consensus. . . . .	97
4-14	Consensus-Adjusted Recommendation Score of three ML-based, two rule-based, and two human predictors when predicting consensus visualization type. Error bars show 95% bootstrapped confidence intervals, with $10^5$ bootstraps. The mean minimum achievable score is the lower dashed line, while the highest achieved CARS is the upper dotted line. . . . .	98
5-1	VizNet enables data scientists and visualization researchers to aggregate data, enumerate visual encodings, and crowdsource effectiveness evaluations. . . . .	102
5-2	Summary statistics (top) and distributions (bottom) of the four source corpora and the VizNet 1M corpus. In the top table, we report the median number of rows and columns. The Distribution column includes the top three most frequent column distributions. Distributions are abbreviated as Norm = normal, L-N = log-normal, Pow = power law, Exp = exponential, Unif = uniform, and Und = undefined. The bottom part of the figure contains distributions describing columns, datasets, and the entire corpus. The bars outlined in red represent three column datasets and the subset which contain one categorical and two quantitative fields. The clustering of three column (C=1, Q=2) datasets is shown in more detail in Figure 5-5. . . . .	106
5-3	VizNet user interface for the <i>Compare Values</i> task experiment. . . . .	107

5-4	Bootstrapped means and 95% confidence intervals for error rates (left) and log response times (right) across tasks and visual encodings for Kim and Heer (2018) original data, and our replication on VizNet. We reuse the original color encoding of Kim and Heer (2018). Shading indicates a statistically significant difference. . . . .	114
5-5	Two-dimensional t-SNE projection of datasets with one categorical and two quantitative columns, evenly sampled from Kim and Heer (2018) and the four corpora within VizNet, with a perplexity of 75. . . . .	116
5-6	Observed log response times (in seconds) vs. those predicted by a gradient boosted regression tree. The dotted diagonal line denotes a perfect relationship between observation and prediction. . . . .	117
5-7	Training $R^2$ and 5-fold cross-validation $R^2$ as the number of training examples increases. . . . .	118
5-8	Performance curves obtained by semi-supervised active learning and supervised learning over 10 iterations. . . . .	122
6-1	Data processing and analysis flow starting from (1) a corpus of real-world datasets, proceeding to (2) feature extraction, (3) mapping from the features to ground truth semantic types from column headers, and then moving to (4) model training and prediction. . . . .	123
6-2	Data types detected by Tableau Desktop 2018.3 for a dataset of country capitals, with and without headers. . . . .	125
6-3	Number of columns per semantic type extracted from VizNet after filtering out the types with less than 1K columns and more than 15% of columns not present in the GloVe dictionary. . . . .	134
6-4	Architecture and hyperparameters of feature-specific subnetworks and primary network. . . . .	135
6-5	Examples of dictionary entries and a learned regular expression for the <b>grades</b> type. . . . .	136
6-6	Rejection curves showing performance while rejecting all but the top $x\%$ highest confidence samples. . . . .	143
7-1	Mean visualization value across three visualizations for crowdsourced workers and experts. . . . .	159

7-2 Relationship between expert and crowdsourced heuristic scores . . . . 160



# List of Tables

2.1	Comparison of machine learning-based visualization recommendation systems. The major differences are that of learning task definition, and the quantity ( $N_{data}$ ) and quality (generation and training data) of the training data. . . . .	37
3.1	Count of participants that corrected their false prior beliefs about the gender wage gap. . . . .	62
3.2	Count of participants that corrected their false prior beliefs about the department with the largest fraction of women. . . . .	62
4.1	Single-column and pairwise-column features used to describe datasets in VizML. . . . .	80
4.2	<b>16 Aggregation functions</b> used to aggregate single- and pairwise-column features into <b>841 dataset-level features</b> . . . . .	81
4.3	Design choice prediction accuracies for five models, averaged over 5-fold cross-validation. The standard error of the mean was $< 0.1\%$ for all results. Results are reported for the neural network (NN) and four baseline models: naive Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR), and random forest (RF). Features are separated into four categories: dimensions (D), types (T), values (V), and names (N). $\mathbf{N}_{raw}$ is the size of the training set before resampling, $\mathbf{d}$ is the number of features, and $C$ is the number of outcome classes. <b>HSA</b> = Has Shared Axis, <b>ISA</b> = Is Shared X-axis or Y-Axis, and <b>XY</b> = Is on X-axis or Y-axis. . . . .	86
4.4	Top-10 feature importances determined by mean decrease impurity for the top performing random forest models. The second column in the visualization-level importances table describes how each feature was aggregated, using the abbreviations in Table 4.2. Colors represent different feature groupings: dimensions (■), type (■), statistical [Q] (■), statistical [C] (■), sequence (■), scale of variation (■), outlier (■), unique (■), name (■), and pairwise-relationship (■). . . . .	88
6.1	Data values sampled from real-world datasets. . . . .	124

6.2	Description of the 27 global statistical features. Asterisks (*) denote features included in Venetis et al. [186]. . . . .	131
6.3	78 semantic types included in this study. . . . .	133
6.4	Support-weighted $F_1$ score, runtime at prediction, and size of Sherlock and four benchmarks. . . . .	139
6.5	Top five and bottom five types by $F_1$ score. . . . .	140
6.7	Examples of low precision and low recall types. . . . .	140
6.9	Performance contribution of isolated feature sets. . . . .	141
6.10	Top-10 features for the decision tree model. “Score” denotes normalized gini impurity. . . . .	142

## Chapter 1

### Introduction

*The ability to take data – to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it – that’s going to be a hugely important skill in the next decades...because now we really do have essentially free and ubiquitous data. So the complementary scarce factor is the ability to understand that data and extract value from it.*

---

*The McKinsey Quarterly*

HAL VARIAN, 2009

Data is increasingly abundant and complex. As of 2012, 2.5 million terabytes of data were generated daily [112]. By 2020, an estimated 1.7 megabytes of data will be generated per person every second [37]. This exponential growth in data is driven by the decreasing cost of data acquisition and storage across industries. Healthcare providers such as Kaiser Permanente generate electronic health records about millions of patient outcomes. Users of social media platforms such as Twitter share over half a billion tweets per day. Commercial enterprises such as Walmart collect petabytes of customer behavior data every hour [77, 85, 112].

The growing supply of data is matched with a commensurate demand for deriving value from data. Across scales – from individuals to organizations to the public sphere – data is increasingly used to improve understanding of systems and the communication of information, in the service of making timely and effective decisions. Embracing data pays off: manufacturing firms that adopt data-driven decision making are statistically more productive, and adoption of data analytics has a positive interaction effect on market performance [17, 38].

Matching the supply of data with the demand for insights requires employing prac-

tices for deriving value from data, such as natural language processing, information retrieval, or machine learning. Data visualization, which leverages the human visual system by encoding information with the visual properties of graphical marks, is one such practice [11, 30, 191]. Researchers have demonstrated the utility of visualization for supporting exploratory data analysis tasks such as detecting patterns and finding outliers [181] and communicative tasks such as authoring narratives [158]. Across industries, from journalism to healthcare to the enterprise, interpreting and authoring visualizations is becoming an essential skill [169].

## 1.1 Problem

Technical users can draw on an abundance of tools for authoring visualizations. Visualization toolkits [46, 59] provide a variety of mechanisms for creating visualizations, such as subclassing a hierarchy of visualization widgets. Visualization grammars, such as D3 and Vega, allow visualization creation through the composition of primitives [15, 154]. Both toolkits and grammars permit the creation of highly customized visualizations and have seen widespread adoption, but they require programming experience. Vector drawing tools like Adobe Illustrator are popular for creating static visualizations, but often have steep learning curves.

Users without programming expertise resort to using tools that have limited flexibility. Visualization tools that are tailored to a specific dataset are increasingly common, but do not let users visualize their own data [148, 165, 205]<sup>123</sup>. Chart typologies [195] such as ManyEyes and charts within Excel let users quickly create visualizations, but restrict users to a small set of chart types [114, 188]. Interactive visualization design tools such as Tableau and Plotly provide drag-and-drop interactions for custom visualization design [130, 173]. However, manually specifying visualizations can be

---

<sup>1</sup><https://nytimes.com/elections/2012/results/president/scenarios.html>

<sup>2</sup><https://projects.fivethirtyeight.com/world-cup-comparisons>

<sup>3</sup><https://ourworldindata.org/grapher/gini-index-around-2015-1990-2015-countries-vs-gini-index-around-1990-1990-2015-countries>

tedious and requires design and analysis expertise, which is not feasible when one is working with a limited amount of time.

As data visualization has begun to be adopted by new communities, the practice has reached users without the technical background, time, or resources to use existing technologies [14, 86]. These users, who are often domain experts like journalists, healthcare practitioners, and business decision-makers, require visualization authoring tools that can be used without the programming expertise or time needed for manual specification. The absence of tools addressing this need leads to users relying on others with technical background, creating “hacky” solutions in existing tools, or not working with data at all.

## 1.2 Approach

The gap between need and capability motivates the guiding question of this thesis: how do we make data visualization systems more accessible to a broader audience, without compromising power? Note that manual specification of visualizations is repetitive because many design decisions are already dictated by visualization best practices and the constraints of the tool being used. As a result, visualization may lend itself to automation. In particular, this thesis focuses on automating components of the visualization workflow through *visualization recommendation*, which automatically identifies and interactively recommends visualizations that are relevant to a specific task and dataset.

Data visualization communicates information by representing data with visual elements. These representations are specified using *encodings* that map from data to the *retinal properties* (e.g., position, length, or color) of *graphical marks* (e.g., points, lines, or rectangles) [11, 20]. For example, to create a scatterplot showing the relationship between fuel efficiency (MPG) and horsepower (Hp) in an automobile dataset, an analyst would encode each pair of data points with the position of a circle on a

2D plane, while also specifying other retinal properties such as size and color:

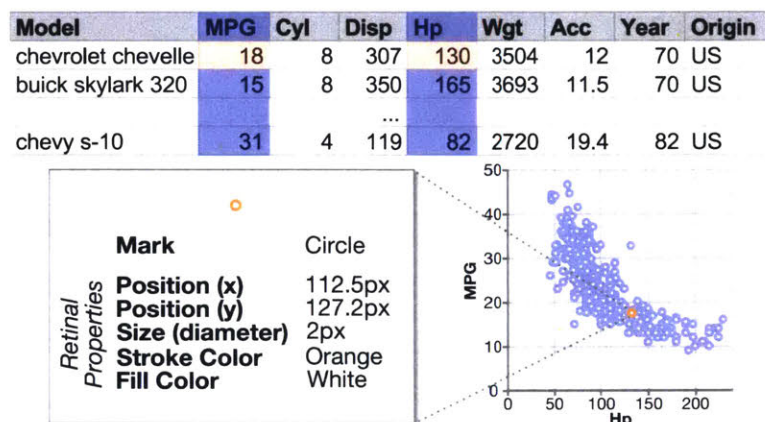


Figure 1-1: Example specification for a 2D scatterplot visualizing two columns of the classic automobile dataset [142].

This simple scatterplot is specified with the Vega-lite [153] grammar by selecting a mark type and fields to be encoded along the x- and y-axes, and in Tableau [173] by placing the two columns onto the respective column and row shelves, as shown in Figure 1-2. But as the complexity of a visualization increases, manually specifying encodings can quickly become prohibitively expensive in terms of time and energy.

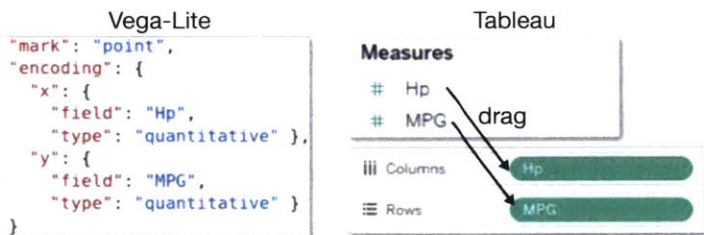


Figure 1-2: Creating a visualization in Vega-Lite and Tableau.

Visualization recommendation aims to reduce the cost of creating visualizations by automatically suggesting the choices that maximize the effectiveness of a given visualization. The effectiveness of a visualization can be defined by informational measures such as efficiency, accuracy, and memorability [13, 208], or emotive measures such as engagement [47, 75]. Prior research also shows that effectiveness is informed by low-level perceptual principles [30, 60, 99, 144] and dataset properties [78, 151], in addition to contextual factors such as the task [6, 78, 150], aesthetics [24], domain [71], audience [163], and medium [116, 156].

Research progress on visualization recommenders demands developments on three complementary but separate fronts of data, methods, and systems. Understanding the relationship between user data, context, task, and visualization effectiveness requires experimental and observational *data*. Modeling the relationship between data and visualization demands work on the *methods* for conceptualizing and developing recommender systems. Finally, presenting recommended results to users requires research on designing the *systems* around recommendations.

### 1.3 Outline and Contributions

This thesis consists of seven chapters, as depicted in Figure 1-3. CHAPTER 2 describes prior work on data visualization systems (§2.1), visualization recommendation (§2.2), data from and for visualization research (§2.3), and semantic type detection (§2.4). This survey of the literature provides context for the four main contributions of the thesis, along with the systems, methods, and data fronts, as described in CHAPTERS 3 through 6.

The challenges of manual specification motivated the creation of **DIVE**, a web-based system that integrates state-of-the-art data exploration features into a single tool. DIVE contributes a mixed-initiative interaction scheme that combines recommendation with point-and-click manual specification, and a consistent visual language that unifies different stages of the data exploration workflow. CHAPTER 3 describes the usage (§3.1), design considerations (§3.2), system design (§3.3), and implementation (§3.4) of DIVE. We evaluated DIVE by conducting a controlled experiment with 67 professional data scientists from two large U.S. consulting firms (§3.5). Our results show that, without prior training, DIVE users were significantly faster than experienced Excel users at completing predefined data visualization and analysis tasks, and were also more likely to correct false prior beliefs. Over 15K users have uploaded 7.5K datasets to DIVE since its release in June 2019. Despite the modest adoption

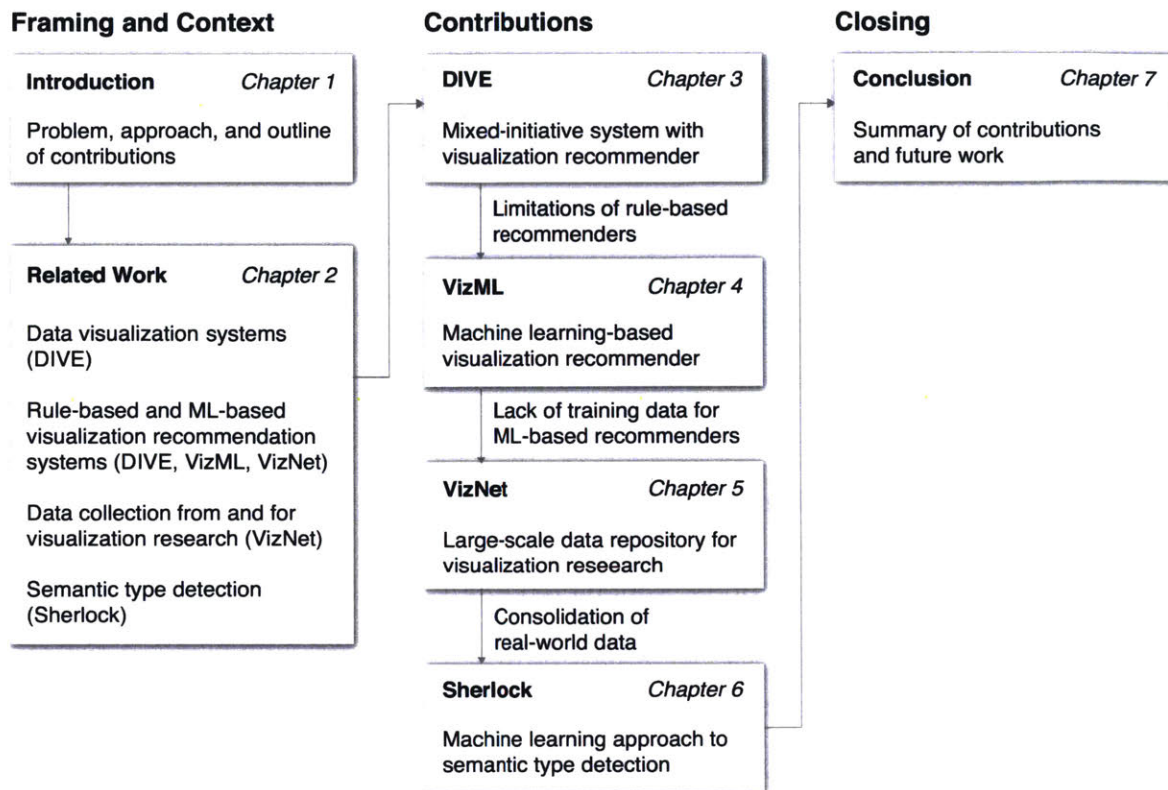


Figure 1-3: Graphical Outline of the Thesis

of DIVE, it was limited by its underlying rule-based recommendation scheme, which did not provide meaningful recommendations for unseen datasets, instead producing an overwhelming quantity of recommendations for multiple selected fields.

The limitations of rule-based recommender systems led to **VizML**, a machine learning-based approach to visualization recommendation using data from a popular online platform. In CHAPTER 4, we first formulate the visualization recommendation problem (§4.1). Then, we identify five key design choices made by analysts while creating visualizations, such as selecting a visualization type and choosing to encode a column along the x- or y-axis. We train models to predict these design choices (§5.2) using one million dataset-visualization pairs collected from a popular online visualization platform (§5.1). Neural networks predict these design choices with a high accuracy when compared with baseline models (§4.4). We report and interpret the importances



of features extracted from one of these baseline models. To evaluate the generalizability and uncertainty of our approach, we benchmark with a crowdsourced test set, and show that the performance of our model is comparable to human performance when predicting consensus visualization type, and exceeds that of other visualization recommender systems (§4.5). While the performance of VizML is promising, the quality of learned models ultimately depends on the quality and quantity of the underlying training data. The data used in current ML-based recommender systems fall short in one of these two dimensions.

The lack of high-quality training data in sufficient quantities motivated the creation of **VizNet**, a large-scale corpus of over 31 million datasets compiled from open data repositories and online visualization galleries. On average, these datasets comprise 17 records over 3 dimensions and across the corpus, and we find 51% of the dimensions record categorical data, 44% quantitative, and only 5% temporal (§5.1). VizNet provides the necessary common baseline for comparing visualization design techniques and developing benchmark models and algorithms for automating visual analysis. To demonstrate VizNet’s utility as a platform for conducting online crowdsourced experiments at scale, we replicate a prior study assessing the influence of user task and data distribution on visual encoding effectiveness, and extend it by considering an additional task: outlier detection (§§5.2, 5.3). To contend with running such studies at scale, we demonstrate how a metric of perceptual effectiveness can be learned from the experimental results and show its predictive power across test datasets (§5.3). By facilitating research with ecologically valid data, VizNet can be useful not only for visualization research, but also for data systems research broadly.

Correctly detecting the semantic type of data columns is crucial for data science tasks such as automated data cleaning, schema matching, and data discovery. The importance of correct semantic types coupled with the data made available through VizNet enabled **Sherlock**, a deep learning approach to semantic type detection. We

train Sherlock on 686,765 data columns retrieved from the VizNet corpus by matching 78 semantic types from DBpedia to the column headers. We characterize each matched column with 1,588 features describing the statistical properties, character distributions, word embeddings, and paragraph vectors of the column values (§6.2). These features are used to train a multi-input neural network (§6.3). The neural network achieves a support-weighted  $F_1$  score of 0.89, exceeding that of a decision tree baseline, dictionary and regular expression benchmarks, and the consensus of crowd-sourced annotations (§6.4). The high performance of our approach, coupled with its apparent robustness to dirty data, encourages future research in using real-world data to train learned data science systems components.

To close, CHAPTER 7 summarizes the contributions of this thesis (§7.1). We discuss limitations of these contributions and recent research developments, before highlighting four promising directions for future research related to visualization recommendation (§7.2). Systems like DIVE require a balance between agency and automation, leading to questions regarding the design of mixed-initiative interfaces, deskilling users, and the interpretability of recommender systems (§7.2.1). As learned systems are adopted in more and more real-world contexts, it is increasingly important to ensure the statistical validity of visual analysis with recommender systems (§7.2.2). The scalability of learned systems like VizML and repositories like VizNet depend on harvesting training data from non-experts, raising open questions about the validity of non-expert judgment compared with expert judgment (§7.2.3).

## 1.4 Prior Publications and Collaboration

This majority of this thesis consists of peer-reviewed work of which I am the primary or co-primary author. But each contribution was the result of a larger effort from many collaborators, especially my advisor Professor César Hidalgo, who supported each project with resources and guidance. DIVE was published at ACM SIGMOD 2018

Workshop on Human-in-the-Loop Data Analytics [66], VizML was published at ACM CHI 2019 [64], VizNet was published at ACM CHI 2019 [65], and Sherlock was published at ACM KDD 2019 [67]. All projects were the result of an large and diverse group of collaborators. To reflect the collective contribution of all contributors, this thesis uses the plural pronoun “we.”

## Chapter 2

### Related Work

This thesis is grounded in four bodies of related work:

1. *Visualization Systems* (§2.1) describes prior systems for visualization creation, authoring, and exploration through manipulation of a graphical interface. Such systems motivate and inform the current thesis in general, and the development of DIVE (CHAPTER 3) in particular.
2. *Visualization Recommenders* (§2.2) surveys rule-based and machine learning-based methods for visualization recommendation. The former rule-based approaches inform the recommendation system underlying DIVE, while both approaches motivate VizML (CHAPTER 4).
3. *Data Collection From and For Visualization Research* (§2.3) reflects on the state of data collected from graphical perception studies. The utility and scarcity of this data prompt the creation of data repositories intended for visualization research, such as VizNet (CHAPTER 5).
4. *Semantic Type Detection* (§2.4) surveys rule-based and data-driven approaches for detecting semantic types. Building on the strengths of prior approaches, Sherlock (CHAPTER 6) is then evaluated against benchmarks exemplifying these approaches.

The goal of this section is to contextualize each thesis contribution by characterizing each body of prior work, identifying gaps in knowledge, and describing the movements in the field.

## 2.1 Visualization Systems

To enable visualization creation, researchers have developed two bodies of prior work. The first consists of programming toolkits and libraries that enable custom visualization design. While programming toolkits and libraries for visualization have seen widespread adoption, this thesis is informed primarily by systems that allow visualization creation through manipulation of a graphical interface.

These systems can be categorized according to dataset specificity and by task. The first category of *dataset-specific* systems 1) investigate new visualization and interaction techniques and 2) enable exploratory analysis. The second category of *dataset-agnostic* systems allow 1) authoring, 2) exploratory analysis, and 3) broadly accessible visualization creation.

### 2.1.1 Dataset-Specific Systems

Data can be used to measure and describe extremely diverse phenomena, from film reviews to baby name trends. Developing systems to visualize these types of heterogeneous inputs is challenging. One way to tackle this heterogeneity is to introduce constraints: *dataset-specific systems* are tailored to specific datasets and, as a result, specific domains. Focusing on specific datasets grants a fixed data model: the space of the objects, attributes, and relationships is fully specified. As a result, dataset-specific systems provide controlled environments for investigating new visualization and interaction techniques as well as for enabling rich exploratory analysis within a chosen domain.

## **Task: Investigating New Visualization and Interaction Techniques**

Seminal data visualization research systems from the mid 1990s such as FilmFinder [5], Table Lens [143], and Seesoft [45] focus on specific data models: film review, tabular baseball statistics, and lines of code data, respectively. By constraining the data model, these “classic” research systems demonstrated the utility of novel visualization and interaction techniques such as tightly coupled query components, direct manipulation, and the integration of context with details. Many of these principles and techniques are now ubiquitous in visualization systems intended to accomplish different tasks for a wider variety of data models.

## **Task: Exploratory Analysis**

Driven by the concurrent rise of personal computing, Internet access, and in-browser visualization libraries, web-based visualizations [59] emerged in the mid 2000s. Name Voyager [192] visualizes the trends of baby names over the past century and throughout all 50 U.S. states. Sense.us [62] enabled collaborative visual analysis of census data. Journalistic outlets such as the *New York Times* graphics department published U.S. election results<sup>1</sup>, Box Office Receipts<sup>2</sup>, and the American Time Use Survey<sup>3</sup>.

As barriers to creating interactive visualizations have decreased, web-based visualizations have become the dominant paradigm for creating, sharing, and consuming interactive visualizations. Data tools are centered around specific datasets and themes, such as the Observatory of Economic Complexity [165] for international trade and DataUSA<sup>4</sup> for US census data. Our World in Data aggregates visualizations and

---

<sup>1</sup><https://www.nytimes.com/elections/2008/results/president/votes.html>

<sup>2</sup>[https://www.nytimes.com/interactive/2008/02/23/movies/20080223\\_REVENUE\\_GRAPHIC.html](https://www.nytimes.com/interactive/2008/02/23/movies/20080223_REVENUE_GRAPHIC.html)

<sup>3</sup><https://www.nytimes.com/interactive/2009/07/31/business/20080801-metrics-graphic.html>

<sup>4</sup><https://datausa.io>

narratives relating to global development time series data. Similarly, Gapminder [50] provides interactive bubble charts to explore development data. Personally, I contributed to Pantheon [205] and the Global Language Network [148], two bespoke visualization systems enabling the exploration of datasets related to historical cultural production and the connections between language groups.

Systems for exploring specific datasets have become indispensable. However, creating such systems requires knowledge of software engineering, a specific data domain, and visualization best practices. As a result, users who wish to visualize their own datasets frequently rely on dataset-agnostic systems that allow users to upload their own data.

### 2.1.2 Dataset-Agnostic Systems

By relaxing the constraint on datasets, *data-agnostic systems* do not make such strict data model assumptions because they accommodate user-uploaded data. Instead, data-agnostic systems are typically constrained to tidy [194] tabular data in which the rows represent observations across the attributes represented by columns. Proceeding from this assumption about semantics – the correspondences between data values and real-world concepts – three categories of data-agnostic systems have emerged.

#### **Task: Accessible Visualization Creation**

Chart typologies let users select from pre-defined visualization types, such as scatterplots and bar charts, and map data values to the encoding channels of these visualizations. A ubiquitous example is the chart template functionality in Microsoft Excel [114]. Similar functionality is present in popular web-based tools such as Web IBM ManyEyes [188], Raw Graphs [111], and Plot.ly Chart Studio [130]. Although chart typologies permit rapid visualizations, they typically support a limited number

of visualization types and customizable parameters.

Instead of constraining users to selecting chart templates, systems based on shelf configuration let users specify encodings through drag-and-drop interactions. Three popular shelf configuration tools include the business intelligence tools tool Tableau (formerly known as Polaris [173]), Spotfire [4], and Qlik Sense [137]. Drag-and-drop interactions are easy to use but do not permit fine-grained specification of a mark style and are still limited in configurability.

### **Task: Authoring**

In response, interactive visual design tools such as Lyra [152], Charticulator [146], and Data Illustrator [100] provide fine-grained control of the visual properties of graphical marks. Built upon data binding models, defined conceptual frameworks such as the Vega specification [154], and constraint-based layout algorithms, these state-of-the-art systems provide the expressiveness needed to create complex visualizations without having to program. Advances in interactive visual design tools address the use case of authoring customized visualizations, while recommender systems facilitate exploring simple visualizations.

### **Task: Exploratory Analysis**

The complete manual specification of visual encodings may be necessary to create customized visualizations. However, for many common use cases, such as preliminary data visualization and creating basic visualizations, the speed and breadth of exploration are more important than customizability. In these cases, visualization recommender systems propose to automatically suggest visualizations for users to search and select, leading to the development of mixed-initiative systems that balance user



interactions with automated recommendation [63].

Prior mixed-initiative systems present a gallery of visualizations with varied underlying data and visual encodings. For example, VizDeck [128] presents users with a ranked list of 1D and 2D visualizations that a user can vote up or down. VizDeck incorporates user votes to update visualization ranks. Small Multiples, Large Singles [183] presents a main visualization and a grid of small multiples that are variants of the main visualization. Users explore recommendations by specifying data queries or visual encodings. Keshif [203] lets users interact with web-based dashboards composed of linked visualizations.

Extending these gallery approaches, recent work utilizes increased computational resources to recommend related views that include unselected fields. Voyager [199] and Voyager 2 [200], along with the underlying Compass recommender engine [198], recommend visualizations involving user-selected fields and one non-selected field. Explore in Google Sheets [52] provides similar recommendations that “look ahead” one field. DIVE is inspired by Voyager 2, and aims to extend its mixed-initiative visualization approach to other parts of the data exploration pipeline. That said, DIVE also extends this line of work by including more non-selected fields in recommendations, incorporating semantic types into recommendations, and introducing a distinction between exact, subset, and expanded recommendations.

## 2.2 Visualization Recommenders

Underlying mixed-initiative visualization systems are visualization recommenders that either suggest data queries (selecting *what* data to visualize) or visual encodings (*how* to visualize the selected data) in response to uploaded data [198]. Recommenders are also categorized into *rule-based* approaches that encode the best practices through hand-crafted rule sets and machine learning-based approaches that learn recommendations directly from data.

### 2.2.1 Data Query Recommenders

Data query recommenders vary widely in their approach. Rank-by-feature framework [159] and Scagnostics [196] present predefined sets of visualizations, such as 2D scatter plots of column pairs, that are ranked by the statistical properties of underlying data. Pre-defined visualizations provide effective overviews of data but can omit meaningful relationships or visualizations. To address this problem of false negatives, Zenvisage [162] and VizDeck [128] enumerate “all possible” visualizations. For example, VizDeck creates 2D scatterplots between all column pairs, which is tractable for small datasets but grows exponentially with the number of columns. In response, recent systems such as SeeDB [107], MuVE [43], and Data Polygamy [27] optimize statistical “utility” functions that serve as proxies for “interestingness” or “relevance.”

Though specifying data queries is crucial to visualization, it is a distinct task from visual encoding recommendation, which is the focus of VizML. Orthogonal to the distinction between data query and visual encoding recommenders is the dimension of rule-based versus machine learning-based systems. Rule-based systems encode visualization best practices as hard-coded *if-then* statements and weights. In contrast, ML-based systems learn the relationship between input data and output visualization.

### 2.2.2 Rule-based Recommenders

Most visual encoding recommenders implement guidelines that are informed by the seminal work of Bertin [11] and Cleveland and McGill [30]. This approach is exemplified by Mackinlay’s APT [105] – the *ur*-recommender system – which enumerates, filters, and scores visualizations using *expressiveness* and perceptual *effectiveness* criteria. The closely related SAGE [149], BOZ [21], and Show Me [106] support more data, encoding, and task types. Recently, hybrid systems such as Voyager [198–200],

Explore in Google Sheets [52,187], and VizDeck [128] have combined visual encoding rules with the recommendation of visualizations that include non-selected columns.

Though effective for many use cases, these systems suffer from three major limitations. First, visualization is a complex process that may require modelling non-linear relationships that are difficult to capture with simple rules. Second, crafting rule sets is a costly process that relies on expert judgment, especially as visualization best practices continue to evolve with research advances. Lastly, as the dimension of input data increases, the combinatorial nature of the rules result in an explosion of possible recommendations.

### 2.2.3 Machine Learning-based Recommenders

The guidelines encoded by rule-based systems are often derived from experimental findings and expert experience. Therefore, through an indirect manner, heuristics distill the best practices learned from another analyst’s experience of creating and consuming visualizations. Instead of aggregating the best practices learned from data and representing them in a system with rules, ML-based systems propose training models that learn directly from data and can be embedded into systems *as is*.

Developed to map JSON-encoded data directly to Vega-Lite visualization specifications, Data2Vis [36] uses a neural machine translation approach to create a sequence-to-sequence model. The model is trained using 4,300 automatically generated Vega-Lite examples, consisting of 1-3 variables that are generated from 11 distinct datasets. Model predictions are qualitatively validated by examining the visualizations generated from 24 common datasets.

Taking a hybrid approach, DeepEye [103] combines rule-based visualization generation with models trained to classify a visualization as “good” or “bad” and rank

lists of these visualizations. The DeepEye corpus consists of 33,412 bivariate visualizations of columns drawn from 42 public datasets. Here, 100 students annotated these visualizations as good/bad, and compared 285,236 pairs. These annotations, combined with 14 features for each column pair, train a decision tree for classification and a ranking neural network [18] for the “learning to rank” task.

Conceptualizing data visualization as a constraint solving problem, Draco-Learn [119] learns trade-offs between the constraints in Draco, a formal model that represents visualizations as logical facts and design guidelines as hard and soft constraints. Constraint weights are learned using a ranking support vector machine trained on ranked pairs of visualizations that are harvested from graphical perception studies [78, 150]. Draco then recommends visualizations that satisfy these constraints by solving a combinatorial optimization problem.

VizML differs from these systems in three major respects. A tabular comparison of the ML-based visualization recommendation systems is shown in Table 2.1. In terms of the *learning task*, DeepEye learns how to classify and rank visualizations, Data2Vis learns an end-to-end generation model, and Draco-Learn learns the weights of soft constraints. By learning to predict design choices, VizML models are easier to quantitatively validate, provide interpretable measures of feature importance, and can be more easily integrated into visualization systems.

In terms of *data quantity*, the VizML training corpus is orders of magnitude larger than that of DeepEye and Data2Vis. The size of our corpus permits the use of 1) large feature sets that capture many aspects of a dataset and 2) high-capacity models such as deep neural networks.

The third major difference is one of *data quality*. In contrast to the few datasets used to train the three existing systems, the datasets used to train VizML models are extremely diverse in shape, structure, and distribution. Furthermore, the visualizations

System	Source	$N_{data}$	Generation	Learning Task	Training Data	Features	Model
VizML	Public (Plotly)	$10^6$	Human	Design Choice Recommendation	Dataset-Visualization Pairs	Single + Pairwise + Aggregated	Neural Network
DeepEye	Crowd	1) 33.4K 2) 285K	Rules $\rightarrow$ Annotation	1) Good-Bad Classif. 2) Ranking	1) Good-Bad Labels 2) Pairwise Comparisons	Column Pair	1) Decision Tree 2) RankNet
Data2Vis	Tool (Voyager)	4,300	Rules $\rightarrow$ Validation	End-to-End Viz. Generation	Dataset Subset-Visualization Pairs	Raw	Seq2Seq NN
Draco-Learn	Crowd	1,100 + 10	Rules $\rightarrow$ Annotation	Soft Constraint Weights	Pairwise Comparisons	Soft Constraint Violation Counts	RankSVM

Table 2.1: Comparison of machine learning-based visualization recommendation systems. The major differences are that of learning task definition, and the quantity ( $N_{data}$ ) and quality (generation and training data) of the training data.

used by other ML-based recommender systems are generated by rule-based systems and evaluated under controlled settings. The corpus used by VizML is the result of real visual analysis by analysts on their own datasets.

However, VizML faces two major limitations. First, these three ML-based systems recommend both data queries and visual encodings, while VizML only recommends the latter. Second, in this paper, we do not create an application that employs our visualization model. Design considerations for user-facing systems that productively and properly employ ML-based visualization recommendations are important, but beyond the scope of this paper.

### 2.3 Data Collection From and For Visualization Research

Rule-based visualization recommender systems encode visualization best practices that are informed by the results of *graphical perception* studies. The data harvested from these studies are increasingly used to train and benchmark learned visualization systems. However, these studies are typically conducted on single datasets with small size and variety, making them difficult to generalize. *Data repositories for visualization research* address the need for centralized data to scale and generalize research, which *machine learning corpora* have facilitated in other domains.

### 2.3.1 Graphical Perception

The visual encoding of data is central to information visualization. Earlier studies have analyzed how different choices of visual encodings such as position, size, color, and shape influence *graphical perception* [30], the decoding of data presented in graphs. Through human subjects experiments, researchers have investigated the effects of visual encoding on the ability to read and make judgments about the data represented in visualizations [30, 58, 82, 92, 164, 167, 168, 179]. Consequently, prior research has provided rankings of visual variables by user performance for nominal, ordinal, and numerical data [30, 92, 104, 105, 161]. Researchers have also studied how design parameters beyond visual encoding variables, such as aspect ratio [28, 57, 176], size [29, 60, 88], chart variation [84, 178], and axis labeling [177], impact the effectiveness of visualizations. Previous studies have evaluated how user task, data types, and distributions influence the effectiveness of charts [150] and visual encoding variables [78].

In current practice, graphical perception experiments are typically conducted on single datasets that are of a small size and variety, lacking the characteristics of real-world data. Studies based on ad hoc datasets may provide useful results but are inherently partial and difficult to generalize, reproduce, and compare against. VizNet provides a corpus of real-world tables from diverse domains to make it easier for researchers to run visualization design evaluation studies at scale. VizNet is sufficiently rich both in size and variety to satisfy the data needs of a substantial number of experimental designs, facilitating the comparison of and reasoning about the results from different experiments on a common baseline.

### **2.3.2 Data Collection for Visualization Research**

Although researchers recognize the need for data collection and generation to facilitate evaluation across a broad range of real datasets [155, 157], little effort has been made to create centralized corpora for data visualization research. Beagle [10] has been used to scrape over 41,000 visualizations from the web. Similarly, the MassVis [13] database was compiled by scraping over 5,000 visualizations from the web and partially annotating them. Lee et al. [160] recently extracted and classified 4.8 million figures from articles on PubMed Central. However, these datasets do not include the raw data represented by the visualizations, limiting their utility for generalized and reproducible visualization research.

### **2.3.3 Machine Learning Corpora**

Recent developments of large-scale data repositories have been instrumental in fostering machine learning research. Access to rich, voluminous data is crucial for developing successful machine learning models and for comparing different approaches using a common baseline. To this end, researchers have created centralized data repositories for training, testing, and benchmarking models across many tasks. Publicly available repositories such as ImageNet [35], SUN [202], COCO [96], and so forth, are one of the main drivers behind the rapid advances in deep learning. VizNet is informed and inspired by the digital experimentation capabilities of large-scale data repositories in machine learning research.

## **2.4 Semantic Type Detection**

Compared with basic atomic types (e.g., strings and numbers), semantic types (e.g., date-time and e-mail address) provide richer and finer-grained descriptions of the

content of a column. Both commercial and research systems have utilized *matching-based* and *data-driven* approaches for semantic type detection.

### 2.4.1 Matching-based

Semantic type detection enhances the functionality of commercial data preparation and analysis systems such as Microsoft Power BI [115], Trifacta [180], and Google Data Studio [53]. To the best of our knowledge, these commercial tools rely on *matching-based* approaches such as manually defined regular expression patterns dictionary lookups of column headers and values to detect a limited set of semantic types. For instance, Trifacta detects around 10 types (e.g., **gender** and **zip code**), and Power BI only supports time-related semantic types (e.g., **date/time** and **duration**). Open source libraries such as messytables [87], datalib [185], and csvkit [54] similarly use heuristics to detect a limited set of types. Benchmarking directly against these systems was unfeasible because of the small number of supported types and lack of extensibility. However, we compare against learned regular expression and dictionary-based benchmarks that are representative of the approaches taken by these systems.

Prior research work with roots in the semantic web and schema-matching literature provides alternative approaches to matching-based type detection. One body of work leverages the existing data on the web, such as WebTables [19], and ontologies (i.e., knowledge bases) such as DBPedia [7], Wikitology [175], and Freebase [12]. Venetis et al. [186] construct a database of value-type mappings and then assign types using a maximum likelihood estimator based on the column values. Syed et al. [175] use column headers and values to build a Wikitology query, the result of which maps columns to types. Informed by these ontology-based approaches, we looked towards existing ontologies to derive the 275 semantic types considered in Sherlock.

Recently Yan and He [204] developed a system that, given a search keyword and



set of positive examples, synthesizes type detection logic from open source GitHub repositories. This system provides a novel approach for leveraging domain-specific heuristics for parsing, validating, and transforming semantic data types. Although both approaches are exciting, the code underlying these systems was not available for benchmarking.

### 2.4.2 Data-driven

Data-driven approaches characterize and compare the statistical properties of data to detect ontology-agnostic semantic types. Ramnandan et al. [141] use heuristics to first separate numerical and textual types and then describe those types using the Kolmogorov-Smirnov (K-S) test and term frequency-inverse document frequency (TF-IDF), respectively. Pham et al. [129] use slightly more features, including the Mann-Whitney test for numerical data and Jaccard similarity for textual data, to train logistic regression and random forest models. We extend these *feature-based* approaches using a significantly larger set of features that includes character-level distributions, word embeddings, and paragraph vectors. Orders of magnitude more features and training samples motivates our use of high-capacity machine learning models such as neural networks. Although the code for benchmarking these models was not available, we include a decision tree model to represent “simpler” machine learning models.

The last category of prior work employs a *probabilistic approach*. Goel et al. [51] use conditional random fields to predict the semantic type of each value within a column, then combine these predictions into a prediction for the whole column. Limaye et al. [94] use probabilistic graphical models to annotate values with entities, columns with types, and column pairs with relationships. These predictions simultaneously maximize a potential function using a message passing algorithm. Probabilistic ap-

proaches are complementary to our machine learning-based approach in that they provide a means for combining column-specific predictions. However, as with prior feature-based models, the code for retraining these models was not made available for benchmarking.

## 2.5 Summary

Visualization systems can be understood in terms of the datasets and tasks they support. As an increasing number of people wish to visualize their own datasets, there has been movement in the field towards accessible systems across a range of expressiveness. Specifically, we describe three visualization tasks: simple chart creation, exploration, and bespoke authoring. This trend towards generalizability has culminated in state-of-the-art interactive visual design systems that provide fine-grained control of encodings and mixed-initiative systems that balance direct manipulation with automation. CHAPTER 3 presents DIVE, one such mixed-initiative system that is based on rule-based visualization recommendation. Recent work mirrors the industry-wide trend towards learned systems by extending rule-based recommenders with machine learning approaches such as VizML (CHAPTER 4). The performance of learned systems are fundamentally constrained by the availability of training data. Although data from graphical perception studies provide a useful starting point, dedicated data repositories such as VizNet (CHAPTER 5) centralize real world datasets for learning and benchmarking. These repositories promote generalizable and scalable visualization research, akin to the effect of machine learning corpora on other domains. But centralized real-world data is useful for data-related research broadly, and has enabled Sherlock (CHAPTER 6), a deep learning approach to semantic type detection.

## Chapter 3

### DIVE

#### *A Mixed-Initiative System Supporting Integrated Data Exploration Workflows*



Figure 3-1: Four stages of a data exploration workflow in DIVE: (A) Datasets, (B) Visualize, (C) Analysis, and (D) Stories.

Knowledge workers across domains – from business to journalism to scientific research – increasingly use data visualization to generate insights, communicate findings, and make decisions [17, 74, 158]. However, many visualization systems, such as those surveyed in CHAPTER 2, rely on manual specification through code [15, 193] or clicks [4, 173].

While required to create bespoke visualizations, manual specification is unnecessary for many common use cases such as preliminary data exploration and the creation of basic visualizations. To support these use cases in which the speed and breadth of exploration are more important than customizability [181], systems can leverage the finding that *the properties of a dataset influence how it can and should be visualized*. For example, prior research has shown that the accuracy with which visual channels (e.g., position and color) encode data depends on the type [11, 30, 191] and distribution [78] of the data values.

To overcome these limitations, visualization recommender systems [107, 128, 199] are now being developed to lower the learning curve of working with data and to facilitate the broad exploration of the result space. Existing recommender systems, however,

do not provide fine-grained control over the results, which impedes the ability to create specific visualizations. This limitation has given rise to hybrid systems [200] with mixed-initiative interfaces [63] that support both broad and focused exploration by combining recommender systems with manual specification.

Yet these mixed-initiative approaches only address isolated parts of an analyst’s data exploration workflow. Ideal workflows involve the multiple stages of identifying the aspects of a dataset that are relevant to questions of interest, bringing a diverse suite of analytical techniques to answer those questions and communicating the results to an audience [181]. In practice, data exploration is a non-linear and iterative process [61] that is often fragmented between multiple tools, even among advanced analysts [74]. Fragmented workflows incur tool and context-switching costs, in addition to the learning costs of each individual tool.

In this chapter, we introduce **DIVE**, a mixed-initiative system combining recommender systems with point-and-click manual specification to support state-of-the-art data model inference, visualization, statistical analysis, and storytelling capabilities. We contribute the design of a system that integrates the multiple stages of the data exploration pipeline, and the description of a system that extends the use of mixed-initiative approaches to data model inference and statistical analysis.

We evaluate DIVE in a controlled user study in which we compared the task performance of analysts using DIVE versus Excel, which supports both data visualization and analysis without requiring prior training. The study involved 67 professionals with significant experience using Excel, but no previous exposure to DIVE. All participants were given the same dataset describing the faculty salaries from a hypothetical university. Then, they reported their prior beliefs (i.e., the expectation) about departments and wages. Next, each participant was randomly assigned to either DIVE or Excel. Finally, after performing six visualization and analysis tasks with the assigned tool, the participants were asked to revise their prior beliefs. Compared with



Figure 3-2: The DIVE user interface. The left navigation bar labeled (A) is used to navigate between modes and modify project properties. The top navigation bar labeled (B) marks the current user state (project, dataset, and mode) and lets users switch projects or datasets. The main pane labeled (C) displays the main results of the mode. The right selection menu (D) lets users change mode-specific parameters (D1) or selecting fields (D2). All other modes in DIVE follow this hierarchical four-section layout, though some modes do not include a selection menu (D).

the Excel users, DIVE users were able to complete more tasks in a shorter amount of time and corrected their false prior beliefs after exploring the data with the tool.

### 3.1 Usage Scenario

We start by describing the example use case of an analyst using DIVE to investigate the factors influencing faculty salary in a hypothetical university. The analyst is provided with an 8 column dataset containing demographic information (name, gender, department, position, and years of experience), measures of performance (number of publications and number of citations), and income (salary) of 1000 faculty members.

She begins by creating a project, after which she is navigated to the **Upload** task associated with her project. Upon successful upload, the **Inspect** task presents her with the inferred data model describing her dataset. This data model consists of *dataset properties*, like dimensions of the dataset, and *field properties*, such as the types and distributions of the fields in her dataset. For example, **name** is correctly detected as a **string** acting as a unique identifier. Here, she can also manipulate the data model by either changing the types of fields, or marking a fields as IDs.

When the analyst is satisfied with her data model, she starts visualizing and analyzing her dataset. Upon clicking the **Visualize** button on the left navigation bar shown in Figure 3-2-B, she is presented with a set of summary visualizations describing the distribution of individual fields. Because she is primarily interested in faculty salaries, she selects the **salary** field on the right hand menu, as seen in Figure 3-2-D. DIVE presents her with visualizations associated with the **salary** field, such as a scatter plot showing a positive relationship between **years of experience** and **salary**, and a box plot showing that **full professors** have a higher median **salary** than **assistant professors**.

The analyst can go beyond visualizations by using the **Analysis** functionality, which supports four types of statistical analysis: construction of **Aggregation** tables, calculation of **Correlation** matrices, **Comparison** of groups through ANOVA, and multivariate **Regression**. First, she enters the Correlation task, which automatically generates a correlation matrix indicating a positive relationship between **salary** and **number of citations** and **number of publications**. She proceeds to the **Regression** task, selects **salary** as her dependent variable, and clicks “Recommend Model.” DIVE recommends a set of models indicating not only the contribution of measures of accomplishment to **salary**, but also the contribution of demographic factors like **gender**.

Finally, the analyst would like to share her results. On the right hand panel of the

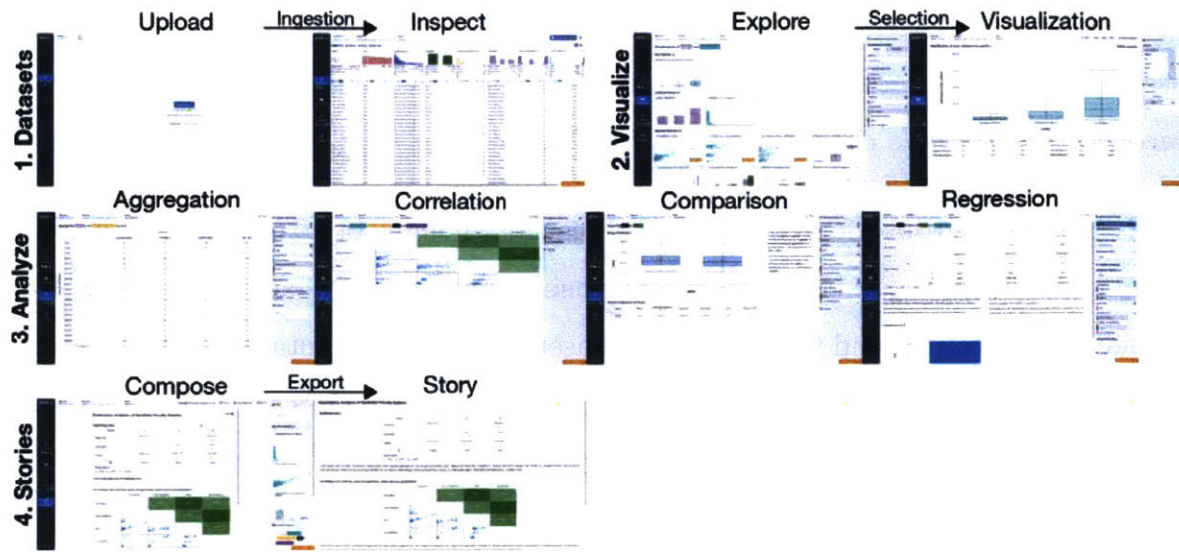


Figure 3-3: Map of the four user stages (rows) and associated tasks (cells) in DIVE. Tasks are connected by an arrow if they are connected by an action. A complete, linear use case of DIVE would begin on the top-left [1. Datasets > Upload] task and progress downwards through modes, towards the bottom-center [4. Stories > Story] task.

**Compose** task, she is able to view thumbnails of the visualizations and analyses she previously saved. By clicking on the thumbnails, she is able to add results to her linear narrative. She can publicly distribute this narrative by sharing an interactive web page tied to the current state of her dataset.

### 3.2 Design Considerations

In this section, we propose a set of four design considerations to guide the design of mixed-initiative systems and multi-stage data exploration workflows.

#### 3.2.1 Discretize Workflow into Tasks Grouped by Ordered Modes

Idealized data exploration workflows consist of a sequential progression of stages flowing from data upload to presentation of results. In practice, analyst workflows

are non-linear and iterative [74]. Furthermore, each stage may consist of multiple discrete but related tasks. *By grouping tasks into ordered stages, we can support non-linear workflows while encouraging a natural progression from data to presentation.*

DIVE is organized into four *modes*, which can be thought of as self-contained, but linked, applications. For example, the **Datasets** mode contains all functionality for uploading, inspecting, and transforming datasets. Each mode contains multiple *tasks*, such as building aggregation tables. The current scheme of modes and tasks is shown in Figure 3-3. Some modes are linked, like the transition from **Upload** to **Inspect** after successful ingestion of an uploaded dataset.

### **3.2.2 Hierarchically Distinguish Between Navigation, Configuration, and Results**

User interface elements either enable users to navigate between tasks and stages, configure inputs to tasks, or view results of a task. Note that these three groupings of elements are dependent, such that tasks determine valid inputs, which then determine results. *Hierarchical visual layouts minimize visual overload while using a uniform visual language across tasks .*

The DIVE interface is organized into four sections, as shown in Figure 3-2. The left navigation bar, shown in every project, (Figure 3-2-A) provides controls for users to navigate between tasks. The top state bar (Figure 3-2-B) shows the user's current project, dataset, and task, and lets users switch between projects and datasets. The task-specific selection menu (Figure 3-2-D) aggregates selectors used to specify or modify inputs to that mode. Results are shown in the main center pane (Figure 3-2-C). The results shown in the center pane are uniquely specified by the state of the selection menu.



### 3.2.3 Organize Input Components into Tightly-Coupled Hierarchies

Within a given task, users work with their data by specifying task configuration as input. *Because task configuration types are inter-dependent, associated interface elements should be hierarchically organized and tightly-coupled [5].*

In DIVE, task input is specified by five types of configuration. Users can specify which model to use, the parameters of a given model, the data used by the model, how to display results, and filter down results based on conditional selectors.

### 3.2.4 Combine Populated Defaults with Incremental Selection

*When a user navigates to a new task without pre-specified configuration, they should be presented results by default, which can then be incrementally modified.* Populated defaults encourage users to engage with their datasets by bypassing the cold-start problem. For example, on the **Explore** page, users are shown descriptive visualizations for each field in their selected dataset. On the **Regression** page, users are shown a set of simple linear regressions involving a random dependent variable and independent variables selected through the LASSO method.

This default result set is modified using the selection menu (Figure 3-2-D), which lets users change the parameters relevant to that specific mode (Figure 3-2-D1) or the field selection (Figure 3-2-D2). As users incrementally select fields, the main view updates to reflect the current state, as shown in Figure 3-4.

### 3.2.5 Distinguish Recommendation Types

As shown in Figure 3-4-B, there is a visual separation between different kinds of results: exact, subset, individual, and expanded. This allows users to both understand

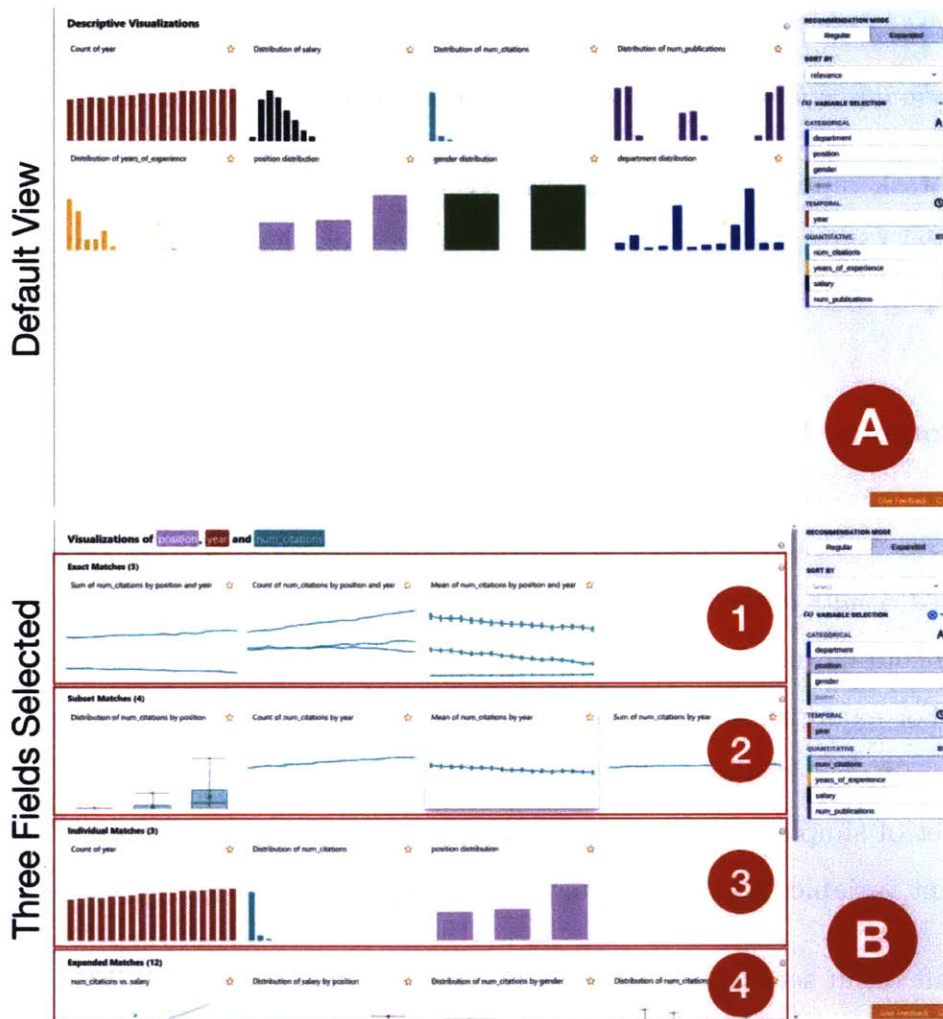


Figure 3-4: Recommended visualizations in the Explore mode, as fields incrementally selected. (A) shows the default view when a user first navigates to Explore, while (B), show recommended visualizations if three fields (**position**, **year**, **salary**) are selected. The sections marked (B1), (B2), (B3), and (B4) contain exact, subset, individual, and expanded matches, respectively.

*how* a recommendation is created and groups results to avoid overwhelming users with recommendations. Additionally, this separation lets users adjust the number of returned results by toggling certain types of recommendations.

### 3.3 System Description

Then, we describe the system features of DIVE, and relate our design choices to the previous design considerations.

#### 3.3.1 Datasets: Upload and Inspect

Meaningful data exploration requires an accurate and relevant data model. For each dataset, DIVE assigned a data model that is comprised of three components: *field properties* such as ID, contiguity, name, semantic type, scale type, statistical properties, and unique values; *inter-field relationships*, including hierarchical relationships between fields; and *inter-object relationships*, like the existence and cardinality of one-to-one, one-to-many, and many-to-many relations. We assume that uploaded data is tabular and *tidy* [194], such that each dataset represents an object, with columns representing attributes of that object and rows representing instances of that object.

**Field Types Definitions.** Following the example of Stevens [172], we distinguish between three general scale types, *nominal*, *ordinal*, and *continuous*, each of which permit specific mathematical transformations and operations. However, similar to Google Data Studio [53], we also distinguish between three general semantic types, *categorical*, *temporal*, and *quantitative*, which inform the families of valid analyses. Each general data type is divided into more specific types, forming a taxonomy of scale and data types.

**Field Type Detection.** DIVE employs a heuristic-based approach for detecting

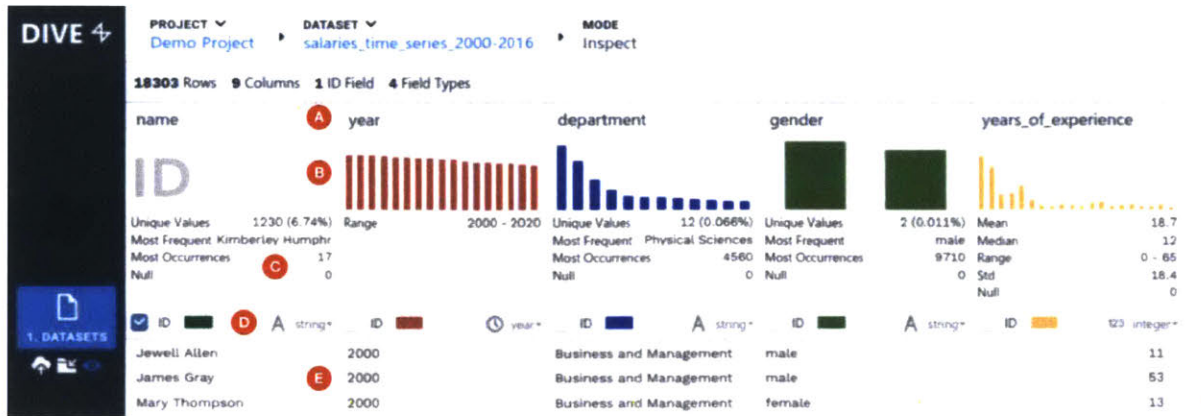


Figure 3-5: Inspect mode table view showing the following inferred field properties: (A) name, (B) ID or univariate descriptive visualization, (C) statistical properties, (D) field color and selectors for changing the color or field type, and (E) data sample.

semantic types, considering both the name of the field and its values. Regular expression matches of names involve comparing the field name against a list of matches and their associated scores. Some semantic types, like `datetime`, involve matching against a set of regular expressions. Others, like `country`, are tested by comparing field values against a list of fixed set of instances. Each field is assigned a score that is normalized by the maximum score for that field across all types. This rank-ordering of types is used to assign confidence scores and suggest field type updates.

**Field Property Detection.** With this detected field type, we can determine the statistical properties of the field. For categorical fields, we determine the number of unique values, and the frequency of each value. For ordinal and continuous fields, we calculate summary statistics. For all data types, we calculate the number of null values, uniqueness, and guess whether a field is an identifier.

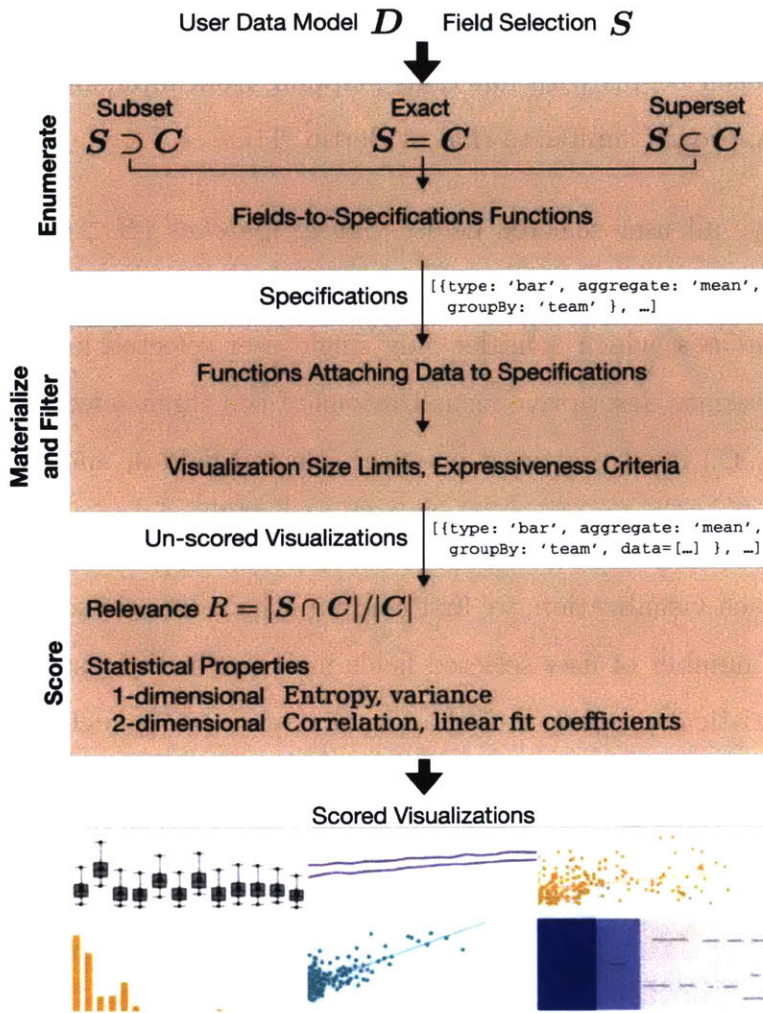


Figure 3-6: Three-stage visualization recommendation system. Given a user data model and user field selection, the system first enumerates visualization specifications, then materializes the specifications and filters based on effectiveness and expressiveness criteria, and finally scores the visualizations.

### 3.3.2 Visualize: Explore and Drill-down

Visualization recommendation in DIVE is a two-stage process starting with *enumeration* of visualization specifications, then *scoring* of visualizations.

Given a user data model  $D$ , user selection  $S = \{s_i\}$ , and un-selected fields  $U = \{u_j\} = D \setminus S$ , our recommendation system iteratively constructs different sets of *considered fields*, denoted as  $C$ . By default, if the user does not select any fields, that is  $S = \{\}$ , DIVE returns univariate descriptive visualizations of each field.

**Enumeration of Visualization Specifications.** We define a *visualization speci-*

*cation* as a statement defining an exact and unique mapping from data to a visualization. Our enumeration approach begins with functions mapping from input fields to specifications, following an approach similar to that of Bertin [11].

*Exact matches* ( $\mathbf{S} = \mathbf{C}$ ) involve all user selected fields. *Subset matches* ( $\mathbf{S} \supset \mathbf{C}$ ) consider a subset of user selected fields, as shown in Figure 3-4-D2. A special case of *subset matches* is *individual matches*, which consider only single user selected fields in  $\mathbf{S}$ . Figure 3-4-D3 shows univariate descriptive visualizations of the three selected fields. *Expanded matches* ( $\mathbf{S} \subset \mathbf{C}$ ) involve at least one user selected field  $s_i$  and at least one un-selected field  $u_j$ , following the approach of [199] and [52].

**Visualization Scoring.** For each visualization, we first compute the *relevance* score  $R = |\mathbf{S} \cap \mathbf{C}|/|\mathbf{C}|$ , marking the number of user-selected fields included in the visualizations. We also compute statistical properties of the visualizations [159], such as entropy, normality, in addition to standard descriptive statistics (min, max, mode, average). For 1-D visualizations. For 2-D visualizations, we compute the correlation and the coefficients of a linear fit. For all visualizations, we attach the number of visual elements and null values.

### 3.3.3 Analysis: Aggregation, Correlation, Comparison, and Regression

DIVE supports four common statistical analysis tasks. By default, analysis tasks are conducted using previously selected user fields. Otherwise, each task has its own procedure for selecting default fields.

The **Aggregation** task encompasses functions for investigating the distribution of groups. This includes creating 1D and 2D contingency tables showing the count of elements in a group, or aggregation tables showing the mean of sum of a value per group. If no fields were selected before, then DIVE selects two fields at random.

The second task, **Correlation**, lets users create correlation matrices containing correlation scatterplots between pairs of quantitative fields. By default, users are presented a 2D contingency table between previously selected fields. If no fields were selected before, then DIVE selects all quantitative fields.

The third task, **Comparison**, lets users compare means of groups using one-way or two-way ANOVA. By default, users are presented a one-way ANOVA with a randomly selected categorical and quantitative field.

The last **Regression** task lets users conduct simple linear or logistic regressions. The results of a model are shown alongside similar models that either leave out single variables or include only one variable. Users can also introduce interaction terms or transform independent variables by taking the log or square. By default, independent variables are chosen by a forward selection algorithm.

### 3.3.4 Stories: Compose and Share

The **Stories** stage lets users assemble saved visualizations, statistical analysis results, and text entries into a linear story. Each result can also be annotated with a title or description. These interactive stories can then be shared with a public URL.

## 3.4 Architecture and Implementation

The architecture of DIVE is diagrammed in Figure 3-7. DIVE is implemented as a web application with front end and back end separation. The front-end uses the React web framework with Redux to manage application state, Google Charts as a visualization library, and Palantir Blueprint as a user interface framework.

The back end consists of a RESTful API using Flask as a web server, a PostgreSQL database for persistence, and Celery on RabbitMQ as an asynchronous task queue.

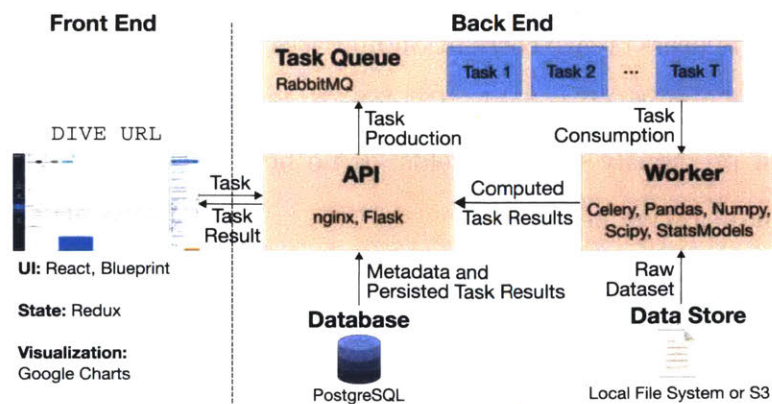


Figure 3-7: DIVE system diagram.

Data manipulation and statistical analysis are performed using the standard Python scientific computing libraries: pandas, NumPy, SciPy, and StatsModels libraries. The server-worker separation enables both asynchronous processing of tasks and scaling through additional worker instances.

### 3.5 Evaluation: DIVE Versus Excel

We conducted a user study in which we asked users to create visualizations and conduct analyses either with DIVE or with Excel, and then compared task performance between these two groups. While Excel is not the state-of-the-art in terms of its functionality, it is still a relevant benchmark for three reasons. First, Excel is one of the most commonly used general tools. This is especially true for "application users," who rely mostly on spreadsheets or dedicated analysis applications such as SAS and SPSS. [74] Second, because we are interested in all steps of the data analysis workflow, we wanted to compare DIVE with a tool supporting both visualization and statistical analysis. At the time of testing, data exploration tools like Tableau, Many Eyes, and Voyager 2 did not support the statistical analyses we tested. Lastly, any other tools would most likely require previous training. In our study, we used a cold-start setting, meaning that participants were not trained in DIVE prior to the experiment.



### 3.5.1 Participant Description

We recruited 75 data analysts from two large consulting firms in the US (19 from company A and 56 from company B) to participate in the experiment. 8 participants were eliminated from the analyses because they did not finish the experiment. Of the remaining 67 participants, 34 participants belonged to the treatment group (DIVE) and 33 to the control group (Excel).

All participants had previous exposure to Excel, averaging 11 years of experience using Excel. None of the participants used DIVE before. Most of the participants (55) use PowerPoint to present their data to their colleagues. The group had a strong technical background, having had taken, on average, 3.94 courses in statistics and 3.49 courses in computer science. 34 participants considered themselves fluent English speakers, 5 proficient, and 28 native. 14 participants had the Bachelor's as the highest degree obtained, 41 had a masters degree and 12 a doctoral degree. Participants reported diverse, but mostly quantitative, undergraduate majors, with the top three majors being Engineering (18), Physics (7), and Economics (7). The average age of our sample was 33.57 years old ( $\sigma^2 = 7.30$  years). There were 12 women in the sample.

### 3.5.2 Experimental Procedure

Participants were given a dataset of faculty salaries from a hypothetical university, which was structurally identical to the dataset described in the **Usage Scenario** section. Each of the 1000 rows corresponds to one faculty member, and the 8 columns correspond to the following fields: **name**, **gender**, **department**, **position**, **years of experience**, **number of publications**, **number of citations**, and **salary**. However, the dataset was constructed to go against general expectations and intuitions. The largest department was the Physical Sciences, with 270 faculty members. Genders

were well distributed across departments, and with women having a higher average salary ( $\mu = \$159,636$ ) than men ( $\mu = \$149,729$ ).

Before receiving the dataset, using a Qualtrics survey, participants reported demographic information (age, gender and native language), education, and software and analytic experience. Next, participants were randomly assigned to the DIVE or the Excel conditions, and were presented with a short description of the dataset. Before asked to complete any tasks, participants were questioned about their expectations regarding the data.

Participants were told that the faculty members in the fictional university belonged to nine different departments, and were asked the following five questions in the **Prior Reporting Section**:

- **[Prior 1]** *Department size*: “Which department do you think has the most professors?” [DEPARTMENT]
- **[Prior 2]** *Fraction of women in department*: “Which department you expect to have the highest fraction of women faculty?” [DEPARTMENT]
- **[Prior 3]** *Average salary by department*: “Which department you expect pays the highest average wage?”
- **[Prior 4]** *Gender wage gap*: “Do you expect there to be a gender wage gap?” [YES | NO]. If they chose YES, they were instructed to indicate whether MEN or WOMEN received higher salaries.
- **[Prior 5]** *Effect of controls on gender wage gap*: “Do you expect the gender wage gap to [INCREASE | DECREASE] when controlling for factors such as years of experience, number of publications, or citations?”

Next, participants were given the salaries dataset, which they uploaded to their assigned tool. Then, they were asked to complete two sections of tasks. In the **Vi-**

**sualization Section**, they were asked to create simple visualizations and paste in Qualtrics the screenshots of the obtained graphs:

- [Visualization 1] *Scatter plot*: “Create a scatter plot of salaries versus number of publications”
- [Visualization 2] *Bar chart of counts*: “Create a bar chart of the number of people by department”
- [Visualization 3] *Bar chart of means*: “Create a bar chart of the average wage by gender”

In **Analysis Section**, participants were asked to answer questions concerning inferential statistics and to paste the screenshots of the evidence that led to their conclusion:

- [Analysis 1] *Difference in means*: “Is the difference between the average wage of males and females statistically significant?” [YES | NO]
- [Analysis 2] *Effect of introducing controls*: “Is the difference between the average wage of males and females statistically significant after controlling for number of citations, publications, department, and years of experience?” [YES | NO]
- [Analysis 3] *Direction of effect after introducing controls*: “Is the difference between the average wage of males and females increased or decreased after controlling for number of citations, publications, department, and years of experience?” [INCREASE | DECREASE].

The final **Prior Updating Section** asked participants to confirm or reject their prior beliefs in light of their obtained results. The main goal of this section was to examine whether users apply the knowledge acquired through the previously performed analyses. In other words, we examined whether they make use of the insight gained to revise their initial expectation about the data.

- **[Prior Update 1]** *Department size*: “Before looking at the data you answered that you expected [selected DEPARTMENT] to have the largest number of faculty. Does the data confirm or reject your hypothesis?” [CONFIRMS | REJECTS]. [If REJECTS] → ”If it rejects, which department has more faculty?”
- **[Prior Update 2]** *Fraction of women in department*: “Before looking at the data you answered that you expected [selected DEPARTMENT] to have the largest fraction of women in the faculty. Does the data confirm or reject your hypothesis?” [CONFIRMS | REJECTS]. [If REJECTS] → ”If it rejects, which department has the largest fraction of women faculty?”
- **[Prior Update 3]** *Average salary by department*: “Before looking at the data you answered that you expected [selected DEPARTMENT] to have the highest average salary. Does the data confirm or reject your hypothesis?” [CONFIRMS | REJECTS]. [If REJECTS] → ”If it rejects, which department has the highest average salary?”
- **[Prior Update 4]** *Gender wage gap*: “Before looking at the data you answered that you expected the data to [SHOW | NOT SHOW] a gender wage gap [If SHOW] → favoring [MEN | WOMEN]?”
- **[Prior Update 5]** *Effect of controls on gender wage gap*: “You also answered that you expected this gender wage gap to [INCREASE | DECREASE] when controlling for factors such as years of experience, number of publications, or citations? Does the data reject or confirm your hypothesis?”

Before leaving the experimental session, DIVE users were asked for feedback regarding their experience in learning how to use DIVE and whether they would use DIVE in the future. Finally, all participants were thanked for their collaboration and debriefed.

### 3.5.3 Experiment Results

Self-assessed familiarity with statistical concepts (from 0, meaning “no experience at all,” to 100, meaning “a lot of experience”) was comparable in the two groups,  $t(65) = 1.07, p = .287$  (DIVE:  $\mu = 64.68, \sigma^2 = 32.37$  — Excel:  $\mu = 55.89, \sigma^2 = 33.90$ ), so any difference in performance cannot not be attributed to differential statistical knowledge in the two groups.

**Prior Reporting Section.** Participants expected the Business and Management department to have the most professors and the highest wages. Liberal Arts and Humanities are expected to be the department with most women faculty. Participants expect to see less women in STEM (Science, Technology, Engineering and Math) departments and more in Social Sciences, Visual Arts and Humanities, and Education.

In response to the wage gap questions, 9 people in this sample responded that they expect no wage gap and 58 responded that they do expect. For the 58 that expect a wage gap, 56 expect men to have a higher wage, whereas only 2 expect women to have higher wage. These results show that most of the participants expect to observe a wage gap between men and women, favoring men. When asked about the wage gap after controlling for other fields, among the participants that said there would be no wage gap, 7 said that any difference between men and women would decrease and 2 indicated that it would increase. From the participants that expected a wage gap, 35 said that the wage gap would decrease after this control, while 23 said it would increase.

**Visualization Section Results:** Participants using DIVE were significantly more successful in creating the visualizations (proportion of YES responses to the question on whether they were able to create the graph):  $\mu = .89, \sigma^2 = .18$ ) than the Excel users ( $\mu = .77, \sigma^2 = .29$ ),  $t(65) = 2.10, p = .04$ .

<i>Tool</i>	<i>Confirm</i>	<i>Don't know</i>	<i>Reject</i>	<i>Marginal Totals</i>
DIVE	6	3	18	27
Excel	8	6	15	29
<i>Marginal Totals</i>	14	9	33	

Table 3.1: Count of participants that corrected their false prior beliefs about the gender wage gap.

<i>Tool</i>	<i>Correct</i>	<i>Incorrect</i>	<i>Marginal Row Totals</i>
DIVE	12	22	34
Excel	4	29	33
<i>Marginal Column Totals</i>	16	51	

Table 3.2: Count of participants that corrected their false prior beliefs about the department with the largest fraction of women.

Next, we looked at the time participants took to upload the graphs for the questions they were able to solve. Participants using DIVE were much faster ( $\mu = 123.50s$ ,  $\sigma^2 = 35.53s$ ) than those using Excel ( $\mu = 168.74s$ ,  $\sigma^2 = 63.59s$ ) at completing the same tasks  $t(64) = 3.60, p = .001$ .

**Analysis Section Results.** For task [Analysis 1] regarding the gender wage gap, of the 34 DIVE users, 13 answered NO, 16 answered YES and only 5 did not answer or selected the I cannot answer this question option. Of the 33 Excel users, 4 answered NO, 17 answered YES and 10 did not answer or selected the I cannot answer this question option.

For task [Analysis 2] regarding the existence of the wage gap after controlling for number of citations, publications, department, and years of experience, 20 DIVE users answered YES, 13 answered NO, and 9 did not answer. Of the Excel group, 16 selected the YES option, 0 selected the NO, and 17 did not answer.

When asked in [Analysis 3] whether the wage gap would increase or decrease after controlling for these factors, 21 DIVE users were able to respond (DECREASE: 6,

INCREASE: 15) while 13 were not able to answer. From the Excel control group, 13 were able to respond (DECREASE: 3, INCREASE: 10) and 20 did not respond. This results suggest that DIVE users reached an answered more often that Excel users. The section does not allow us to verify the accuracy of the answers, which is assessed in the next section.

**Prior Updating Results.** All DIVE users corrected their answer to the question regarding the largest department (32 out of 34 participants, the other two where correct from the beginning and did not change their answer). 28 Excel user corrected their response to the same question (28 out of 33), whereas 3 did not correct and two were correct from the beginning. Thus, the two groups do not differ in the amount of correction for this question (*Fisher's exact test* = .26,  $p > .05$ ).

For the question regarding the department with most women both DIVE users and Excel users had stereotypical prior beliefs, in the sense that they did not expect STEM department to have many women. None of DIVE users selected STEM department, and only one Excel user selected Engineering. The correct department in this dataset is Physical Science, that has a total of 122 women. After completing the visualization tasks, more participants corrected their prior when using DIVE than when using Excel. This result suggests that the type of tool used significantly affects the correction of false prior beliefs (*Fisher's exact test* = .04,  $p < .05$ ).

Of the participants that expected a wage gap favoring men, 18 DIVE users rejected their priors (18 out of 27; 6 confirmed and 3 said I do not know). 15 Excel users (15 out of 29; 8 confirmed and 6 said I do not know) rejected their prior expectations for this same questions. The amount of rejection is not significantly different in the two groups (*Fisher's exact test* = .29,  $p > .05$ ).

### 3.5.4 Qualitative Participant Feedback

At the end of the hour-long session, participants assigned to the DIVE group had a chance to give free-form text feedback in response to the question **”What features of DIVE did you like most?”** Many participants commented on the ease with which they could create visualizations using DIVE and the ease to get started:

*“**Very quick** to produce visualizations, fairly **instinctive** in use [sic]”* *“How **easy** it was to visualize information across different charts”* *“Easy to **get started**”* *“Intuitive to **learn** with some stats and data analysis background”*

Some participants commented specifically on the visualization recommendation:

*“**Proactive** graph proposal”* *“**Automation** [sic] creation of the basic data visualizations”* *“**Automatic** analysis when choosing variables”*

Others appreciated the integration of visualization and statistical analysis:

*“**integrated place** for many tasks”* *“The ubiquity of visualization as part of the analysis.”*

### 3.5.5 Experimental Results Summary

The performance of data analysts using DIVE and using Excel was compared under controlled conditions. The choice of comparing DIVE with Excel was motivated by the popularity of Excel for all stages of data exploration. In our sample, all participants had previously used Excel and none of them had previous experience using DIVE. The experimental results suggest that participants using DIVE were more successful and faster than those using Excel in completing the same data exploration tasks. DIVE users also corrected false prior beliefs more often than Excel users. These results



suggest that that DIVE supports learning from data by facilitating the completion of visualization and analysis tasks.

### **3.6 Summary and Future Work**

We introduce DIVE, a mixed-initiative data exploration tool originally developed to lower the learning curve to working with data. We described the heuristic-based approaches for data model inference and visualization recommendation system that is central to the construction of DIVE. We also described the considerations taken into account in the interface, interaction, and system design of workflows that integrate the multiple stages of data exploration.

While DIVE is intentionally a domain-agnostic system, it is an open question whether developing domain-specific approaches would be more tractable and powerful. For example, data tools for survey data could prioritize clustering by demographic, while tools for financial data could build in forecasting and candlestick charts.

We also plan to address problems of scalability by permitting synchronous batch processing ahead of time, such that visualizations and analyses are pre-computed. This would be most relevant for a frequently used dataset, especially by a team. This approach would be required for a model describing which recommendations should be calculated and presented first. It would also require global measures for scoring and ranking results.

There remains significant future work that must be done to improve the components of data exploration systems. Data model inference in existing tools is largely ad hoc and built from scratch. There is significant space for the contribution of principled systems for data model inference, on top of which others can build tools. Improvements to state-of-the-art systems could train a machine learning algorithm on a corpus of annotated datasets, and interaction-driven approaches could be semi-automated,

with users incrementally updating the inferred types. There is also an opportunity to improve upon existing rule-based systems for visualization recommendation. Our current system does not support the transformation of fields or combining fields, which would significantly increase the size of recommended visualizations. Analogous systems could also be developed to recommend statistical analyses.

## Chapter 4

# VizML

### *A Machine Learning Approach to Visualization Recommendation*

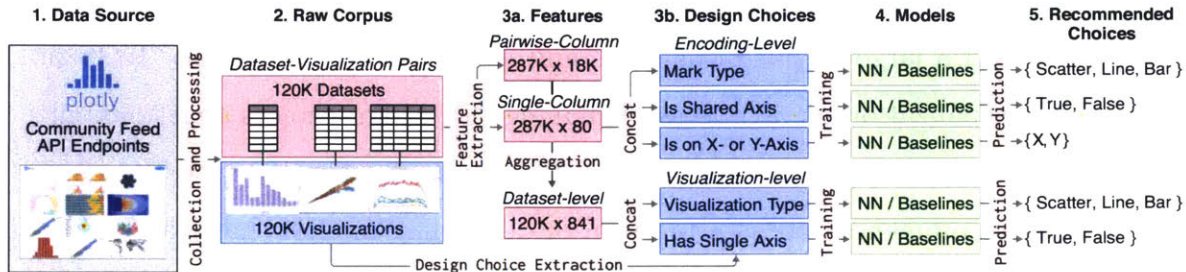


Figure 4-1: A diagram of the data processing and analysis flow in VizML, starting from (1) the original Plotly Community Feed API endpoints, proceeding to (2) the deduplicated dataset-visualization pairs, (3a) features describing each individual column, pair of columns, and dataset, (3b) the design choices extracted from visualizations, (4) task-specific models trained on these features, and (5) potential recommended design choices.

Many visualization recommender systems, including DIVE, encode visualization guidelines as a collection of “if-then” statements, or *rules* [55], to automatically generate visualizations for analysts to search and select from, rather than manually specify [184]. For example, APT [105], BOZ [21], and SAGE [149] generate and rank visualizations using rules informed by perceptual principles. Recent systems such as Voyager [199, 200], Show Me [106], and DIVE [66] extend these approaches with support for column selection. While effective for certain use cases [199], these *rule-based* approaches face limitations such as costly rule creation and the combinatorial explosion of possible results [2].

In contrast, *machine learning (ML)-based* systems directly learn the relationship between data and visualizations by training models using analyst interaction. While recent systems such as DeepEye [103], Data2Vis [36], and Draco-Learn [119] are exciting, they do not learn how to make visualization design choices as an analyst would,

which impacts the interpretability and ease of integration into existing systems. Furthermore, because these systems are trained with annotations on rule-generated visualizations in controlled settings, they are limited by the quantity and quality of data.

In this chapter we introduce **VizML**, a ML-based approach to visualization recommendation using a large corpus of datasets and associated visualizations. To begin, we describe visualization as a process of making the design choices that maximize effectiveness, which depends on the dataset, task, and context. Then, we formulate visualization recommendation as a problem of developing models that learn how to make design choices.

We train and test machine learning models using one million unique dataset-visualization pairs from the Plotly Community Feed [132]. We describe our process of collecting and cleaning this corpus, extracting features from each dataset, and extracting five key design choices from the corresponding visualizations. Our learning task is to optimize models that use the features of datasets to predict these choices.

Neural networks trained on 60% of the corpus achieve  $\sim 70-95\%$  accuracy at predicting design choices in a separate 20% test set. This performance exceeds that of four simpler baseline models, which themselves out-perform random chance. We report feature importances from one of these baseline models, interpret the contribution of features to a given task, and relate them to existing research.

We evaluate the generalizability and uncertainty of our model by benchmarking it against a crowdsourced test set. We construct this test set by randomly selecting datasets from Plotly, visualizing each as a bar, line, and scatter plot, and then measuring the consensus of Mechanical Turk workers. Using a scoring metric that adjusts for the degree of consensus, we find that VizML performs comparably to Plotly users and Mechanical Turkers, and outperforms two rule-based and two ML-based visual-

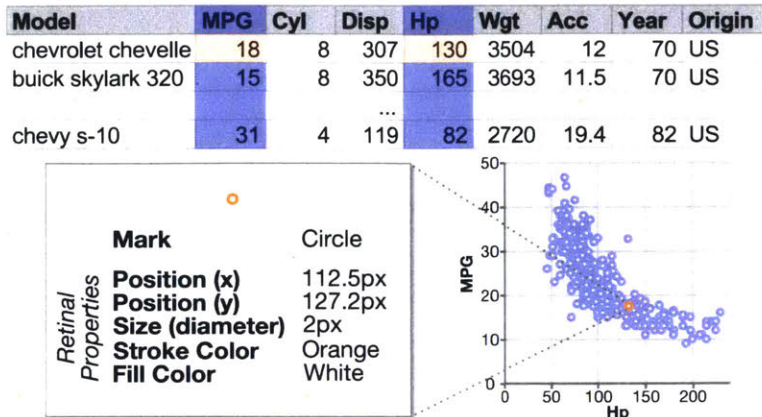


Figure 4-2: Example specification for a 2D scatterplot visualizing two columns of the Cars dataset.

ization recommendation systems.

To conclude, we discuss interpretations, applications, and limitations of our initial machine learning approach to visualization recommendation. We also suggest directions for future research, such as aggregating public training and benchmarking corpora, integrating separate recommender models into an end-to-end system, and refining the definitions of visualization effectiveness.

#### 4.1 Problem Formulation

Data visualization communicates information by representing data with visual elements. These representations are specified using *encodings* that map from data to the *retinal properties* (e.g. position, length, or color) of *graphical marks* (e.g. points, lines, or rectangles) [11, 20].

Concretely, consider a dataset that describes 406 automobiles (rows) with eight attributes (columns) such as miles per gallon (MPG), horsepower (Hp), and weight in pounds (Wgt) [142]. To create a scatterplot showing the relationship between MPG and Hp, an analyst encodes each pair of data points with the position of a circle on a 2D plane, while also specifying other retinal properties such as size and color.

To create bespoke visualizations, analysts may need to exhaustively specify encodings

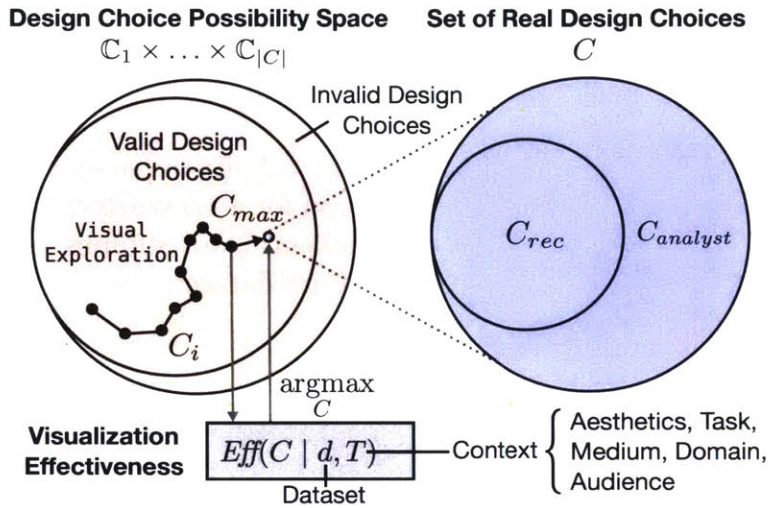


Figure 4-3: Creating visualizations is a process of making design choices, which can be recommended by a system or specified by an analyst.

in detail using expressive tools. But a scatterplot is specified with the Vega-lite [153] grammar by selecting a mark type and fields to be encoded along the x- and y-axes, and in Tableau [173] by placing the two columns onto the respective column and row shelves.

That is, to create basic visualizations in many grammars or tools, an analyst specifies higher-level *design choices*, which we define as statements that compactly and uniquely specify one or more lower-level encodings. Equivalently, each grammar or tool affords a design space of visualizations, which a user constrains by making choices.

We formulate basic visualization of a dataset  $d$  as a set of interrelated design choices  $C = \{c\}$ , each of which is selected from a possibility space  $c \sim \mathcal{C}$ . However, not all design choices result in valid visualizations – some choices are incompatible with each other. For instance, encoding a categorical column with the Y position of a line mark is invalid. Therefore, the set of choices that result in valid visualizations is a subset of the space of all possible choices  $C_1 \times C_2 \times \dots \times C_{|C|}$ .

The effectiveness of a visualization can be defined by informational measures such as efficiency, accuracy, and memorability [13, 208], or emotive measures like engagement [47, 75]. Prior research also shows that effectiveness is informed by low-level

perceptual principles [30, 60, 99, 144] and dataset properties [78, 151], in addition to contextual factors such as task [6, 78, 150], aesthetics [24], domain [71], audience [163], and medium [116, 156]. In other words, an analyst makes design choices  $C_{max}$  that maximize visualization effectiveness  $Eff$  given a dataset  $d$  and contextual factors  $T$ :

$$C_{max} = \arg \max_C Eff(C | d, T) \quad (4.1)$$

But making design choices can be expensive. A goal of visualization recommendation is to reduce the cost of creating visualizations by automatically suggesting a subset of design choices  $C_{rec} \subseteq C$  that maximize effectiveness. Trained with a corpus of datasets  $\{d\}$  and corresponding design choices  $\{C\}$ , ML-based recommender systems treat recommendation as an optimization problem, such that predicted  $C_{rec} \sim C_{max}$ .

#### 4.1.1 Modeling Design Choice Recommendation

Consider a single design choice  $c \in C$ . Let  $C' = C \setminus \{c\}$  denote the set of all other design choices excluding  $c$ . Given  $C'$ , a dataset  $d$ , and context  $T$ , there is an ideal design choice recommendation function  $F_c$  that outputs the design choice  $c_{max} \in C_{max}$  from Eqn. 4.1 that maximizes visualization effectiveness:

$$F_c(d | C', T) = c_{max} \quad (4.2)$$

Our goal is to approximate  $F_c$  with a function  $G_c \approx F_c$ . Assume now a corpus of datasets  $D = \{d\}$  and corresponding visualizations  $V = \{V_d\}$ , each of which can be described by design choices  $C_d = \{c_d\}$ . Machine learning-based recommender systems consider  $G_c$  as a model with a set of parameters  $\Theta_c$  that can be trained on this corpus by a learning algorithm that maximizes an objective function  $Obj$ :

$$\Theta_{fit} = \arg \max_{\Theta_c} \sum_{d \in D} Obj(c_d, G_c(d | \Theta_c, C', T)) \quad (4.3)$$

Without loss of generality, say the objective function maximizes the likelihood of observing the training output  $\{C_d\}$ . Even if an analyst makes sub-optimal design choices, collectively optimizing the likelihood of all observed design choices can still be optimal [121]. This is precisely the case with our observed design choices  $c_d = F_c(d | C', T) + \text{noise} + \text{bias}$ . Therefore, given an unseen dataset  $d^*$ , maximizing this objective function can plausibly lead to a recommendation that maximizes effectiveness of a visualization.

$$G_c(d^* | \Theta_{fit}, C', T) \approx F_c(d^* | C', T) = c_{max} \quad (4.4)$$

In this paper, our model  $G_c$  is a neural network and  $\Theta_c$  are connection weights. We simplify the recommendation problem by optimizing each  $G_c$  independently, and without contextual factors:  $G_c(d | \Theta) = G_c(d | \Theta, C', T)$ . We note that independent recommendations may not be compatible, nor do they necessarily maximize overall effectiveness. Generating a complete visualization output will require modeling dependencies between  $G_c$  for each  $c$ .

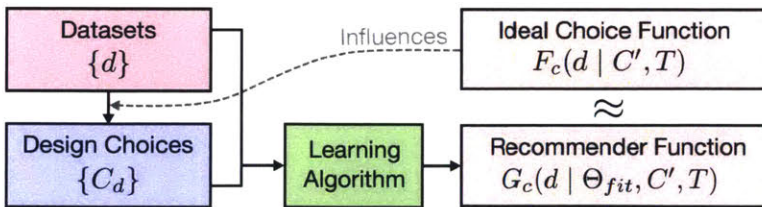


Figure 4-4: Basic setup of learning models to recommend design choices with a corpus of datasets and corresponding design choices.

## 4.2 Data

We describe our process for extracting features and design choices from the processed Plotly data. These are steps **1**, **2** and **3** in Figure 4-1. We describe our process for



collecting and cleaning the corpus of 2.3 million dataset-visualization pairs from the Plotly Community Feed [130,132] and provide a description of the data. This paper is the first time the *Plotly corpus*, generated by 143,007 unique users, is used to train visualization recommender systems. The corpus along with analysis scripts is publicly available at <https://vizml.media.mit.edu>.

#### 4.2.1 Collection and Cleaning

**Plotly** [130] is a software company that creates tools and software libraries for data visualization and analysis. For example, Plotly Chart Studio [131] is a web application that lets users upload datasets and manually create interactive D3.js and WebGL visualizations of over 20 visualization types. Users familiar with Python can use the Plotly Python library [133] to create those same visualizations with code.

Visualizations in Plotly are specified with a declarative schema. In this schema, each visualization is specified with two data structures. The first is a list of *traces* that specify how a collection of data is visualized. The second is a dictionary that specifies aesthetic aspects of a visualization untied from the data, such as axis labels and annotations. For example, the scatterplot from Section 4.1 is specified with a single “scatter” trace with Hp as the x parameter and MPG as the y parameter:

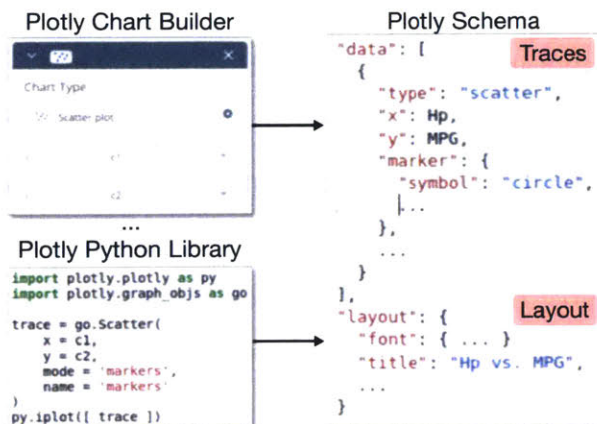


Figure 4-5: Creating a scatterplot in the Plotly schema using the Plotly Chart Builder and the Plotly Python Library.

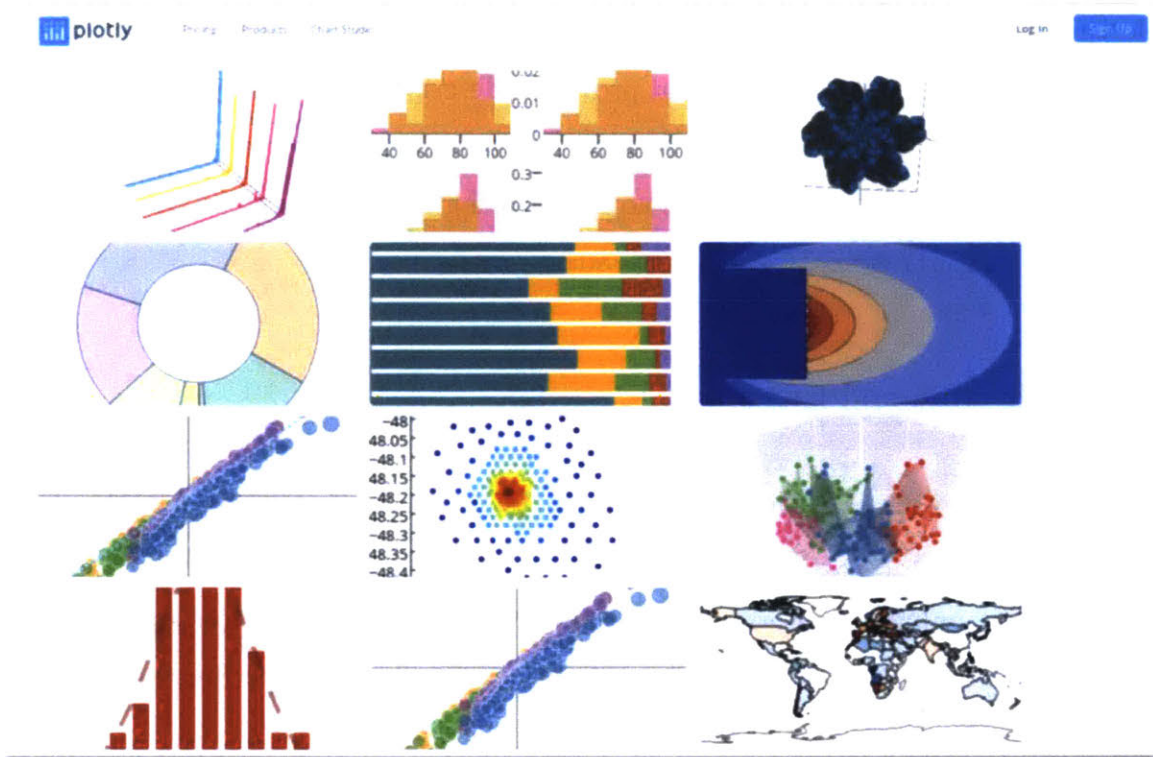


Figure 4-6: Screenshot of the Plotly Community Feed [132].

The Plotly schema is similar to that of MATLAB and of the matplotlib Python library. The popular Vega [154] and Vega-lite [153] schemas are more opinionated, which “allows for complicated chart display with a concise JSON description, but leaves less control to the user” [135]. Despite these differences, it is straightforward to convert Plotly schemas into other schemas, and vice versa.

Plotly also supports sharing and collaboration. Starting in 2015, users could publish charts to the Plotly Community Feed [132], which provides an interface for searching, sorting, and filtering millions of visualizations, as shown in Figure 4-6. The underlying `/plots` endpoint from the Plotly REST API [134] associates each visualization with three objects: `data` contains the source data, `specification` contains the traces, and `layout` defines display configuration.

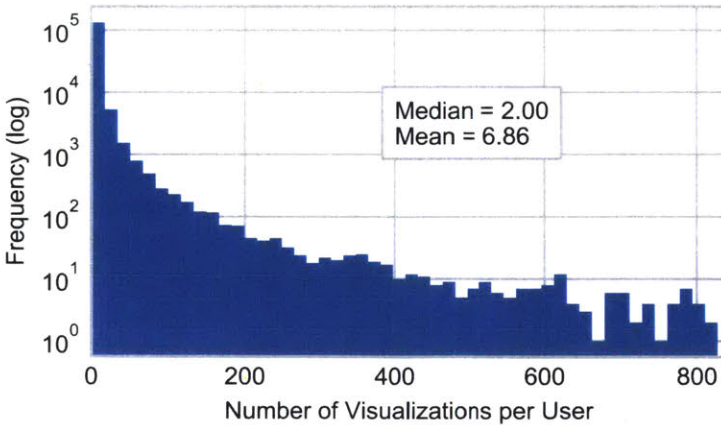


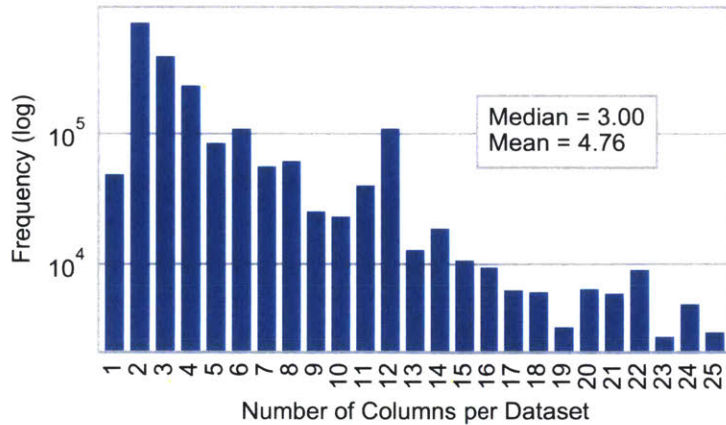
Figure 4-7: Distribution of plots per user, visualized on a log-linear scale.

#### 4.2.2 Data Description

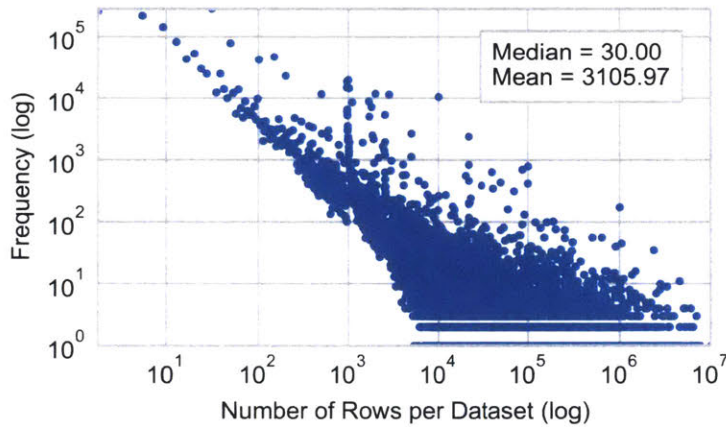
Using the Plotly API, we collected approximately 2.5 years of public visualizations from the feed, starting from 2015-07-17 and ending at 2018-01-06. We gathered 2,359,175 visualizations in total, 2,102,121 of which contained all three configuration objects, and 1,989,068 of which were parsed without error. To avoid confusion between user-uploaded datasets and our dataset of datasets, we refer to this collection of dataset-visualization pairs as the *Plotly corpus*.

The Plotly corpus contains visualizations created by 143,007 unique users, who vary widely in their usage. The distribution of visualizations per user is shown in Figure 4-7. Excluding the top 0.1% of users with the most visualizations, many of whom are bots that programmatically generate visualizations, users created a mean of 6.86 and a median of 2 visualizations each.

Datasets also vary widely in number of columns and rows. Though some datasets contain upwards of 100 columns, 94.97% contain less than or equal to 25 columns. Excluding datasets with more than 25 columns, the average dataset has 4.75 columns, and the median dataset has 3 columns. The distribution of columns per visualization is shown in Figure 4-8a. The distribution of rows per dataset is shown in Figure 4-8b,



(a) Distribution of columns per dataset, after removing the 5.03% of datasets with more than 25 columns, visualized on a log-linear scale.



(b) Distribution of rows per dataset, visualized on a log-log scale.

Figure 4-8: Distribution of dataset dimensions in the Plotly corpus.

and has a mean of 3105.97, median of 30, and maximum of  $10 \times 10^6$ . These heavy-tailed distributions are consistent with those of IBM ManyEyes and Tableau Public as reported by [120].

Though Plotly lets users generate visualizations using multiple datasets, 98.32% of visualizations used only one source dataset. Therefore, we are only concerned with visualizations using a single dataset. Furthermore, over 90% of visualizations used all columns in the source dataset, so we are not able to address data query selection. Lastly, out of 13, 321, 598 traces, only 0.16% of have transformations or aggregations. Given this extreme class imbalance, we are not able to address column transformation

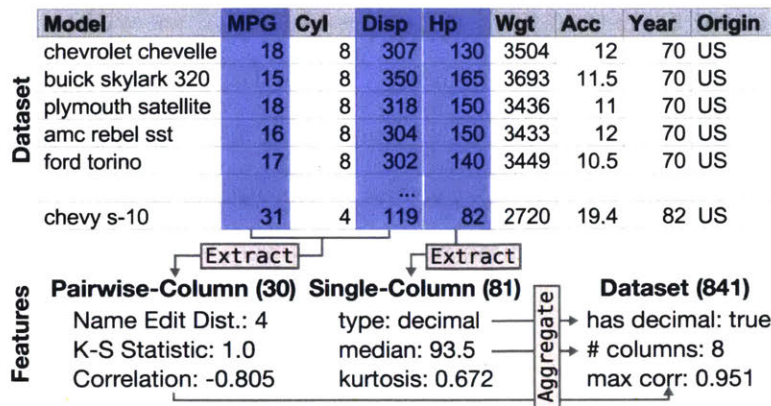


Figure 4-9: Extracting features from the Automobile MPG dataset. [142]

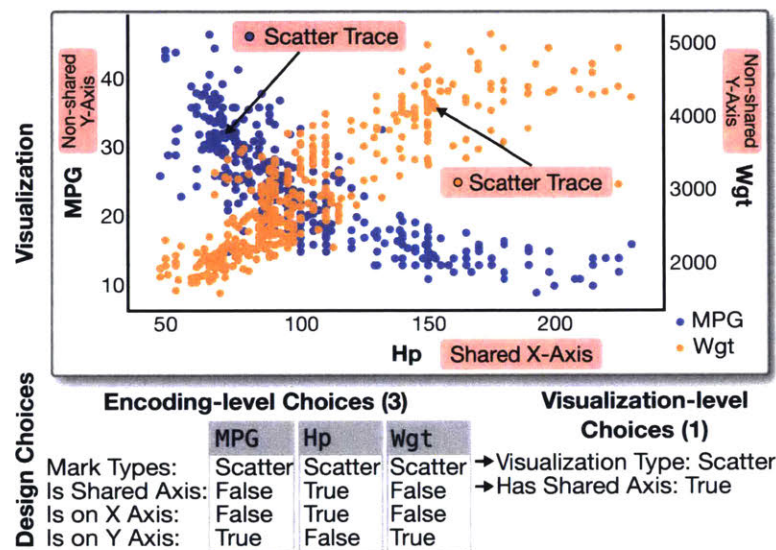


Figure 4-10: Extracting design choices from a dual-axis scatterplot visualizing three columns of the MPG dataset.

or aggregation as learning tasks.

### 4.2.3 Feature Extraction

We map each dataset to 841 features, mapped from 81 single-column features and 30 pairwise-column features using 16 aggregation functions. Detail on each of the features is found in Tables 4.1 and 4.2.

Each column is described by **81 single-column features** across four categories. The **Dimensions (D)** feature is the number of rows in a column. **Types (T)** features capture whether a column is categorical, temporal, or quantitative. **Values**

(**V**) features describe the statistical and structural properties of the values within a column. **Names (N)** features describe the column name. We distinguish between these feature categories for three reasons. First, these categories let us organize how we create and interpret features. Second, we can observe the contribution of different types of features. Third, some categories of features may be less generalizable than others. We order these categories (**D**  $\rightarrow$  **T**  $\rightarrow$  **V**  $\rightarrow$  **N**) by how biased we expect those features to be towards the Plotly corpus.

We describe each pair of columns with **30 pairwise-column features**. These features fall into two categories: **Values** and **Names**. Note that many pairwise-column features depend on the individual column types determined through single-column feature extraction. For instance, the Pearson correlation coefficient requires two numeric columns, and the “number of shared values” feature requires two categorical columns.

We create **841 dataset-level features** by aggregating these single- and pairwise-column features using the **16 aggregation functions** shown in Table 4.2. These aggregation functions convert single-column features (across all columns) and pairwise-column features (across all pairs of columns) into scalar values. For example, given a dataset, we can count the number of columns, describe the percent of columns that are categorical, and compute the mean correlation between all pairs of quantitative columns. Two other approaches to incorporating single-column features are to train separate models per number of columns, or to include column features with padding. Neither approach yielded a significant improvement over the results reported in Section 4.4.

Details on the 81 single-column features, 30 pairwise-column features and 16 aggregation functions can be found in Tables 4.1. Single-column features fall into four categories: **Dimensions (D)** (number of rows in a column), **Types (T)** (categorical, temporal, or quantitative), **Values (V)** (the statistical and structural properties)

and **Names (N)** (related to column name). Pairwise-column features fall into two categories” **Values** and **Names**. Finally, 841 dataset-level features are created by aggregating these features using the **16 aggregation functions** shown in Table 4.2.

#### 4.2.4 Design Choice Extraction

Each visualization in Plotly consists of traces that associate collections of data with visual elements. Therefore, we extract an analyst’s design choices by parsing these traces. Examples of **encoding-level design choices** include *mark type*, such as scatter, line, bar; and *X or Y column encoding*, which specifies which column is represented on which axis; and whether or not an X or Y column is the single column represented along that axis. For example, the visualization in Figure 4-10 consists of two scatter traces, both of which have the same column encoded on the X axis (**Hp**), and two distinct columns encoded on the Y axis (**MPG** and **Wgt**).

By aggregating these encoding-level design choices, we can characterize **visualization-level design choices** of a chart. Within our corpus, over 90% of the visualizations consist of homogeneous mark types. Therefore, we use *visualization type* to describe the type shared among all traces, and also determined whether the visualization *has a shared axis*. The example in Figure 4-10 has a scatter visualization type and a single shared axis (X).

### 4.3 Methods

We describe our feature processing pipeline, the machine learning models we use, how we train those models, and how we evaluate performance. These are steps **4** and **5** of the workflow in Figure 4-1.

(a) **81 single-column features** describing the dimensions, types, values, and names of individual columns.

Dimensions (1)	
Length (1)	Number of values
Types (8)	
General (3)	Categorical (C), quantitative (Q), temporal (T)
Specific (5)	String, boolean, integer, decimal, datetime
Values (58)	
Statistical [Q, T] (16)	Mean, median, range $\times$ (Raw/normalized by max), variance, standard deviation, coefficient of variance, minimum, maximum, (25th/75th) percentile, median absolute deviation, average absolute deviation, quantitative coefficient of dispersion
Distribution [Q] (14)	Entropy, Gini, skewness, kurtosis, moments (5-10), normality (statistic, p-value), is normal at ( $p < 0.05$ , $p < 0.01$ ).
Outliers (8)	(Has/%) outliers at ( $1.5 \times \text{IQR}$ , $3 \times \text{IQR}$ , 99%ile, $3\sigma$ )
Statistical [C] (7)	Entropy, (mean/median) value length, (min, std, max) length of values, % of mode
Sequence (7)	Is sorted, is monotonic, sortedness, (linear/log) space sequence coefficient, is (linear/space) space
Unique (3)	(Is/#!/%) unique
Missing (3)	(Has/#!/%) missing values
Names (14)	
Properties (4)	Name length, # words, # uppercase characters, starts with uppercase letter
Value (10)	("x", "y", "id", "time", digit, whitespace, "\$", "€", "£", "¥") in name

(b) **30 pairwise-column features** describing the relationship between values and names of pairs of columns.

Values (25)	
[Q-Q] (8)	Correlation (value, $p$ , $p < 0.05$ ), Kolmogorov-Smirnov (value, $p$ , $p < 0.05$ ), (has, %) overlapping range
[C-C] (6)	$\chi^2$ (value, $p$ , $p < 0.05$ ), nestedness (value, = 1, $> 0.95\%$ )
[C-Q] (3)	One-Way ANOVA (value, $p$ , $p < 0.05$ )
Shared values (8)	is identical, (has/#!/%) shared values, unique values are identical, (has/#!/%) shared unique values
Names (5)	
Character (2)	Edit distance (raw/normalized)
Word (3)	(Has, #, %) shared words

Table 4.1: Single-column and pairwise-column features used to describe datasets in VizML.



Categorical (5)	Number (#), percent (%), has, only one (#=1), all
Quantitative (10)	Mean, variance, standard deviation, coefficient of variance (CV), min, max, range, normalized range (NR), average absolute deviation (AAD), median absolute deviation (MAD)
Special (1)	Entropy of data types

Table 4.2: **16 Aggregation functions** used to aggregate single- and pairwise-column features into **841 dataset-level features**.

### 4.3.1 Feature Processing

We converted raw features into a form suitable for modeling using a five-stage pipeline. First, we apply one-hot encoding to categorical features. Second, we set numeric values above the 99th percentile or below the 1st percentile to those respective cut-offs. Third, we imputed missing categorical values using the mode of non-missing values, and missing numeric values with the mean of non-missing values. Fourth, we removed the mean of numeric fields and scaled to unit variance.

Lastly, we randomly removed datasets that were exact deduplicates of each other, resulting in unique 1,066,443 datasets and 2,884,437 columns. However, many datasets are slight modifications of each other, uploaded by the same user. Therefore, we removed all but one randomly selected dataset per user, which also removed bias towards more prolific Plotly users. This aggressive deduplication resulted in a final corpus of **119,815 datasets** and **287,416 columns**. Results from only exact deduplication result in significantly higher within-corpus test accuracies, while a soft threshold-based deduplication results in similar test accuracies.

### 4.3.2 Prediction Tasks

Our task is to train models that use the features described in Section 4.2.3 to predict the design choices also described in Section 4.2.4. **Two visualization-level predic-**

**tion tasks** use dataset-level features to predict visualization-level design choices:

1. **Visualization Type [VT]: 2-, 3-, and 6-class**

Given all traces are the same type, what type is it?

<i>Scatter</i>	<i>Line</i>	<i>Bar</i>	<i>Box</i>	<i>Histogram</i>	<i>Pie</i>
44829	26209	16002	4981	4091	3144

2. **Has Shared Axis [HSA]: 2-class**

Do the traces all share one axis (either X or Y)?

<i>False</i>	<i>True</i>
95723	24092

The **three encoding-level prediction tasks** use features about individual columns to predict how they are visually encoded. These prediction tasks consider each column independently, instead of alongside other columns in the same dataset, which accounts for the effect of column order.

1. **Mark Type [MT]: 2-, 3-, and 6-class**

What mark type is used to represent this column?

<i>Scatter</i>	<i>Line</i>	<i>Bar</i>	<i>Box</i>	<i>Histo</i>	<i>Heatmap</i>
68931	64726	30023	13125	5163	1032

2. **Is Shared X-axis or Y-axis [ISA]: 2-class**

Is this column the only column encoded on its axis?

<i>False</i>	<i>True</i>
275886	11530

3. **Is on X-axis or Y-axis [XY]: 2-class**

Is this column encoded on the X-axis or the Y-axis?

<i>False</i>	<i>True</i>
144364	142814

For the **Visualization Type** and **Mark Type** tasks, the 2-class task predicts line vs. bar, and the 3-class predicts scatter vs. line vs. bar. Though Plotly supports over twenty mark types, we limited prediction outcomes to the few types that comprise the majority of visualizations within our corpus. This heterogeneity of visualization types is consistent with the findings of [10, 120].

### 4.3.3 Neural Network and Baseline Models

Our primary model is a fully-connected feedforward neural network (NN) with 3 hidden layers, each consisting of 1,000 neurons with ReLU activation functions and implemented using PyTorch [124]. For comparison, we chose four simpler baseline models, all implemented using scikit-learn [125] with default parameters: naive Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR) and random forest (RF). Randomized parameter search for each model did not result in a significant performance increase over the reported results.

For all models, we split the data into 60/20/20 train/validation/test sets and train and test each model five times using 5-fold cross-validation. The reported results are thus test results averaged across the five test sets. We oversample the train, validation, and test sets to the size of the majority class while ensuring no overlap between the three sets. We oversample because of the heterogeneous outcomes, naive classifiers guessing the base rates would have high accuracies. Balanced classes also allow us to report standard accuracies (fraction of correct predictions), ideal for interpretability and generalizing results to multi-class cases  $C > 2$ , in contrast to measures such as the  $F_1$  score.

The neural network was trained with the Adam optimizer and a mini-batch size of 200. The learning rate was initialized at  $5 \times 10^{-4}$ , and followed a learning rate schedule that reduces the learning rate by a factor of 10 upon encountering a plateau,

defined as 10 epochs during which validation accuracy does not increase beyond a threshold of  $10^{-3}$ . Training ended after the third decrease in the learning rate, or at 100 epochs. Weight decay, dropout and batch normalization did not significantly improve performances.

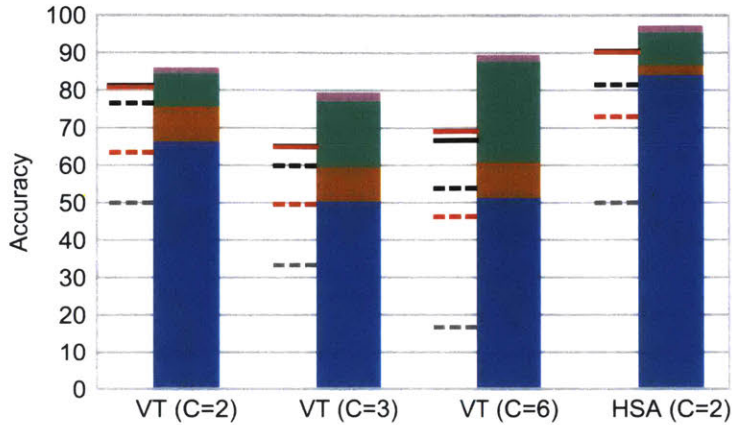
In terms of features, we constructed four different feature sets by incrementally adding the **Dimensions (D)**, **Types (T)**, **Values (V)**, and **Names (N)** categories of features, in that order. We refer to these feature sets as **D**, **D+T**, **D+T+V**, and **D+T+V+N=All**. The neural network was trained and tested using all four feature sets independently. The four baseline models only used the full feature set (**D+T+V+N=All**).

#### 4.4 Evaluating Performance

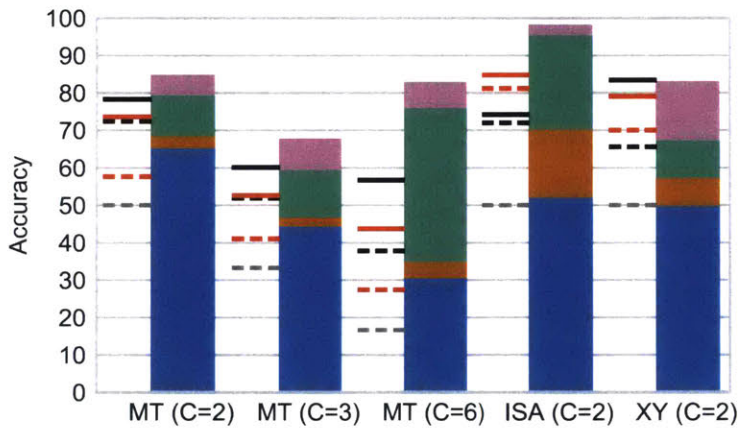
We report performance of each model on the five prediction tasks in the barplot in Figure 4-11 and in Table 4.3. The neural network consistently outperforms the baseline models and model performance generally progressed as  $\text{NB} < \text{KNN} < \text{LR} \approx \text{RF} < \text{NN}$ . That said, the performance of both RF and LR is not significantly lower than that of the NN in some cases. Simpler classifiers may be desirable, depending on the need for optimized accuracy, and the trade-off with other factors such as interpretability and training cost.

Because the four feature sets are a sequence of supersets ( $D \subset D+T \subset D+T+V \subset D+T+V+N$ ), we consider the accuracy of each feature set above and beyond the previous. For instance, the increase in accuracy of a model trained on **D+T+V** over a model trained on **D+T** is a measure of the contribution of value-based (**V**) features. These marginal accuracies are visualized alongside baseline model accuracies in Figure 4-11.

We note that the value-based feature set (*e.g.* the statistical properties of a column)



(a) Marginal accuracies by feature set for visualization-level prediction tasks.



(b) Marginal accuracies by feature set for encoding-level prediction tasks.



Figure 4-11: Marginal contribution to neural network accuracy by feature set, for each task. Baseline accuracies are shown as solid and dashed lines for naive Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR), and random forest (RF). **HSA** = Has Shared Axis, **ISA** = Is Shared X-axis or Y-axis and **XY** = Is on X-axis or Y-axis.

(a) Prediction accuracies for two visualization-level tasks.

Model	Features	d	Visualization Type			HSA
			$C=2$	$C=3$	$C=6$	$C=2$
NN	D	15	66.3	50.4	51.3	84.1
	D+T	52	75.7	59.6	60.8	86.7
	D+T+V	717	84.5	77.2	87.7	95.4
	All	841	<b>86.0</b>	<b>79.4</b>	<b>89.4</b>	<b>97.3</b>
NB	All	841	63.4	49.5	46.2	72.9
KNN	All	841	76.5	59.9	53.8	81.5
LR	All	841	<b>81.8</b>	64.9	<b>69.0</b>	90.2
RF	All	841	81.2	<b>65.1</b>	66.6	<b>90.4</b>
$N_{\text{raw}}$ (in 1000s)			42.2	87.0	99.3	119

(b) Prediction accuracies for three encoding-level tasks.

Model	Features	d	Mark Type			ISA	XY
			$C=2$	$C=3$	$C=6$	$C=2$	$C=2$
NN	D	1	65.2	44.3	30.5	52.1	49.9
	D+T	9	68.5	46.8	35.0	70.3	57.3
	D+T+V	66	79.4	59.4	76.0	95.5	67.4
	All	81	<b>84.9</b>	<b>67.8</b>	<b>82.9</b>	<b>98.3</b>	<b>83.1</b>
NB	All	81	57.6	41.1	27.4	81.2	70.0
KNN	All	81	72.4	51.9	37.8	72.0	65.6
LR	All	81	73.6	52.6	43.7	<b>84.8</b>	79.1
RF	All	81	<b>78.3</b>	<b>60.1</b>	<b>46.7</b>	74.2	<b>83.4</b>
$N_{\text{raw}}$ (in 1000s)			94.7	163	183	287	287

Table 4.3: Design choice prediction accuracies for five models, averaged over 5-fold cross-validation. The standard error of the mean was  $< 0.1\%$  for all results. Results are reported for the neural network (NN) and four baseline models: naive Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR), and random forest (RF). Features are separated into four categories: dimensions (D), types (T), values (V), and names (N).  $N_{\text{raw}}$  is the size of the training set before resampling,  $d$  is the number of features, and  $C$  is the number of outcome classes. **HSA** = Has Shared Axis, **ISA** = Is Shared X-axis or Y-Axis, and **XY** = Is on X-axis or Y-axis.

contribute more to performance than the type-based feature set (*e.g.* whether a column is categorical), potentially because there are many more value-based features than type-based features. Or, because many value-based features are dependent on column type, there may be overlapping information between value- and type-based features.

#### 4.4.1 Interpreting Feature Importances

Feature importances help relate our results to prior literature and inform design guidelines for rule-based systems. Here, we determine feature importances for our top performing random forest models using the standard mean decrease impurity (MDI) measure [16, 102]. We choose this method for its interpretability and its stability across runs. The top ten features for five different tasks are shown in Table 4.4a.

We first note the importance of **dimensionality** (■), like the length of columns (*i.e.* the number of rows) or the number of columns. For example, the length of a column is the second most important feature for predicting whether that column is visualized as a line or bar trace. The dependence of mark type on number of visual elements is consistent with heuristics like “keep the total number of bars under 12” for showing individual differences in a bar chart [166], and not creating pie charts with more “more than five to seven” slices [83]. The dependence on number of columns is related to the heuristics described by Bertin [11] and encoded in Show Me [106].

Features related to **column type** (■) are consistently important for each prediction task. For example, whether a dataset contains a string type column is the fifth most important feature for determining two-class visualization type. The dependence of visualization type choice on column data type is consistent with the type-dependency of the perceptual properties of visual encodings described by Mackinlay [105] and Cleveland and McGill [30].

(a) Feature importances for two visualization-level tasks.

#	Visualization Type ( $C=2$ )	Has Shared Axis ( $C=2$ )
1	% Values are Mode	std
2	Min Value Length	max
3	Entropy	var
4	Entropy	std
5	String Type	has
6	Median Length	max
7	Mean Value Length	AAD
8	Entropy	mean
9	Entropy	max
10	Min Value Length	AAD

(b) Feature importances for three encoding-level tasks.

#	Mark Type ( $C=2$ )	Is Shared Axis ( $C=2$ )	Is X or Y Axis ( $C=2$ )
1	Entropy	# Words In Name	Y In Name
2	Length	Unique Percent	X In Name
3	Sortedness	Field Name Length	Field Name Length
4	% Outliers (1.5IQR)	Is Sorted	Sortedness
5	Field Name Length	Sortedness	Length
6	Lin Space Seq Coeff	X In Name	Entropy
7	% Outliers (3IQR)	Y In Name	Lin Space Seq Coeff
8	Norm. Mean	Lin Space Seq Coeff	Kurtosis
9	Skewness	Min	# Uppercase Chars
10	Norm. Range	Length	Skewness

Table 4.4: Top-10 feature importances determined by mean decrease impurity for the top performing random forest models. The second column in the visualization-level importances table describes how each feature was aggregated, using the abbreviations in Table 4.2. Colors represent different feature groupings: dimensions (■), type (■), statistical [Q] (■), statistical [C] (■), sequence (■), scale of variation (■), outlier (■), unique (■), name (■), and pairwise-relationship (■).

**Statistical features** (quantitative: ■, categorical: ■) such as Gini, entropy, skewness and kurtosis are important across the board. The presence of these higher order moments is striking because lower-order moments such as mean and variance are low in importance. The importance of these moments highlight the potential importance of capturing high-level characteristics of distributional shape. These observations sup-



port the use of statistical properties in visualization recommendation, like in [159,197], but also the use of higher-order properties such as skewness, kurtosis, and entropy in systems such as Foresight [34], VizDeck [128], and Draco [119].

**Measures of orderedness** (■), specifically sortedness and monotonicity, are important for many tasks. Sortedness is defined as the element-wise correlation between the sorted and unsorted values of a column, that is  $|corr(\mathbf{X}_{raw}, \mathbf{X}_{sorted})|$ , which lies in the range  $[0, 1]$ . Monotonicity is determined by strictly increasing or decreasing values in  $\mathbf{X}_{raw}$ . The importance of these features could be due to pre-sorting of a dataset by an analyst, which may reveal which column is considered to be the independent or explanatory column, which is typically visualized along the X-axis. While intuitive, we have not seen orderedness factor into existing systems.

We also note the importance of the linear or logarithmic space sequence coefficients, which are heuristic-based features that roughly capture the **scale of variation** (■). Specifically, the linear space sequence coefficient is determined by  $std(\mathbf{Y})/mean(\mathbf{Y})$ , where  $\mathbf{Y} = \{\mathbf{X}_i - \mathbf{X}_{i-1}\}$  with  $i = (1 + 1)..N$  for the linear space sequence coefficient, and  $\mathbf{Y} = \{\mathbf{X}_i/\mathbf{X}_{i-1}\}$  with  $i = (1 + 1)..N$  for the logarithmic space sequence coefficient. A column “is” linear or logarithmic if its coefficient  $\leq 10^{-3}$ . Both coefficients are important in all four selected encoding-level prediction tasks. We have not seen similar measures of scale used in prior systems.

In sum, the diversity of the features in Table 4.4a suggest that rule-based recommender systems should include more features than the current type based features most systems rely on (*e.g.* [106,200]). Furthermore, the task-specific ranking of features, as well as the non-linear dependencies in the models, make it even harder for rule-based systems to perform well across tasks and domains and thus further emphasize the need for ML-based recommender systems

## 4.5 Benchmarking with Crowdsourced Effectiveness

We expand our definition of effectiveness from a binary to a continuous function that can be determined through crowdsourced consensus. Then, we describe our experimental procedure for gathering visualization type evaluations from Mechanical Turk workers. We compare different models at predicting these evaluations using a consensus-based effectiveness score.

### 4.5.1 Modeling and Measuring Effectiveness

As discussed in Section 4.1, we model data visualization as a process of making a set of design choices  $C = \{c\}$  that maximize an effectiveness criteria  $Eff$  that depends on dataset  $d$ , task, and context. In Section 4.4, we predict these design choices by training a machine learning model on a corpus of dataset-design choice pairs  $[(d, c_d)]$ . But because each dataset was visualized only once by each user, we consider the user choices  $c_d$  to be effective, and each other choice as ineffective. That is, we consider effectiveness to be binary.

But prior research suggests that effectiveness is continuous. For example, Saket et al. use time and accuracy preference to measure task performance [150], Borkin et al. use a normalized memorability score [13], and Cleveland and McGill use absolute error rates to measure performance on elementary perceptual tasks [30]. Discussions by visualization experts [72, 81] also suggest that multiple visualizations can be equally effective at displaying the same data.

Our effectiveness metric should be continuous and reflect the ambiguous nature of data visualization, which leads to multiple choices receiving a non-zero or even maximal score for the same dataset. This is in agreement with measures of performance for other machine learning tasks such as the BLEU score in language translation [123]

and the ROUGE metric in text summarization [95], where multiple results can be (partly) correct.

To estimate this effectiveness function, we need to observe a dataset  $d$  visualized by multiple potential users. Assume that a design choice  $c$  can take on multiple discrete values  $\{v\}$ . For instance, we consider  $c$  the choice of **Visualization Type**, which can take on the values  $\{bar, line, scatter\}$ . Using  $n_v$  to denote the number of times  $v$  was chosen, we compute the probability of making choice  $v$  as  $\hat{P}_c(v) = n_v/N$ , and use  $\{\hat{P}_c\}$  to denote the collection of probabilities across all  $v$ . We normalize the probability of choice  $v$  by the maximum probability to define an effectiveness score  $\hat{Eff}_c(v) = \hat{P}_c(v) / \max(\{\hat{P}_c\})$ .

$$\hat{Eff}_c(v) = \hat{P}_c(v) / \max(\{\hat{P}_c\}) \quad (4.5)$$

Now, if all  $N$  users make the same choice  $v$ , only  $c = v$  will get the maximum score while every other choice  $c \neq v$  will receive a zero score. However, if two choices are chosen with an equal probability and are thus both equally effective, the normalization will ensure that both receive a maximum score.

Developing this crowdsourced score that reflects the ambiguous nature of making data visualization choices serves three main purposes. First, it lets us establish uncertainty around our models – in this case, by bootstrap. Second, it lets us test whether models trained on the Plotly corpus can generalize and if Plotly users are actually making optimal choices. Lastly, it lets us benchmark against performance of the Plotly users as well as other predictors.

To generate the crowdsourced evaluation data, we recruited and successfully pre-screened 300 participants through Amazon Mechanical Turk. The data preparation

and crowdsourced evaluation procedures is described in more detail in the next two sections.

#### 4.5.2 Data Preparation

To select the datasets in our benchmarking test set, we first randomly surfaced a set of candidate datasets that were visualized as either a bar, line, or scatter chart. Then, we removed obviously incomplete visualizations (*e.g.* blank visualizations). Finally, we removed datasets that could not be visually encoded in all three visualization types without losing information. From the remaining set of candidates, we randomly selected 33 bar charts, 33 line charts, and 33 scatter charts.

As we cleaned the data, we adhered to four principles: modify the user’s selections as little as possible, apply changes consistently to every dataset, rely on Plotly defaults, and don’t make any change that is not obvious. For each of these datasets, we modified the raw column names to remove Plotly-specific biases (*e.g.* removing “,**x**” or “,**y**” that was automatically append to column names). We also wanted to make the user evaluation experience as close to the original chart creation experience as possible. Therefore, we changed column names from machine-generated types if they are obvious from the user visualization axis labels or legend (*e.g.* the first column is unlabeled but visualized as **Sepal Width** on the X-axis). Because of these modifications, both the Plotly users and the Mechanical Turkers accessed more information than our model.

We visualized each of these 99 datasets as a bar, line, and scatter chart. We created these visualizations by forking the original Plotly visualization then modifying Mark Types using Plotly Chart Studio. We ensured that color choices and axis ranges were consistent between all visualization types. The rest of the layout was held constant to the user’s original specification, or the defaults provided by Plotly.

## 4.6 Crowdsourced Evaluation Procedure

For the crowdsourced evaluation, we recruited participants through Amazon Mechanical Turk. To participate in the experiment, workers had to hold a U.S. bachelor degree and be at least 18 years of age, and be completing the survey on a phone. Workers also had to successfully answer three prescreen questions: 1) Have you ever seen a data visualization? [**Yes** or No], 2) Does the x-axis of a two-dimensional plot run horizontally or vertically? [**Horizontally**, Vertically, Both, Neither], 3) Which of the following visualizations is a bar chart? [**Picture of Bar Chart**, Picture of Line Chart, Picture of Scatter]. 150 workers successfully completed the two-class experiment, while 150 separate workers completed the three-class experiment.

After successfully completing the pre-screen, workers evaluated the visualization type of 30 randomly selected datasets from our test set. Each evaluation had two stages. First, the user was presented the first 10 rows of the dataset, and told to "Please take a moment to examine the following dataset. (Showing first 10 out of X rows)." Then, after five seconds, the "next" button appeared. At the next stage, the user was asked "Which visualization best represents this dataset? (Showing first 10 out of X rows)." On this stage, the user was shown both the dataset and the corresponding bar, line, and scatter charts representing that dataset. A user could submit this question after a minimum of ten seconds. The evaluations were split into two groups of 15 by an attention check question. Therefore, each of the 66 datasets were evaluated 68.18 times on average, while each of the 99 ground truth datasets was evaluated 30 times on average.

Users also answered the question "How confident are you in your answer?", with a five-point scale ranging from "Not at all confident" to "Completely confident."

At the end of the survey, participants were asked "How easy were the previous questions?", "How prepared did you feel to answer the previous questions?" and "Do you

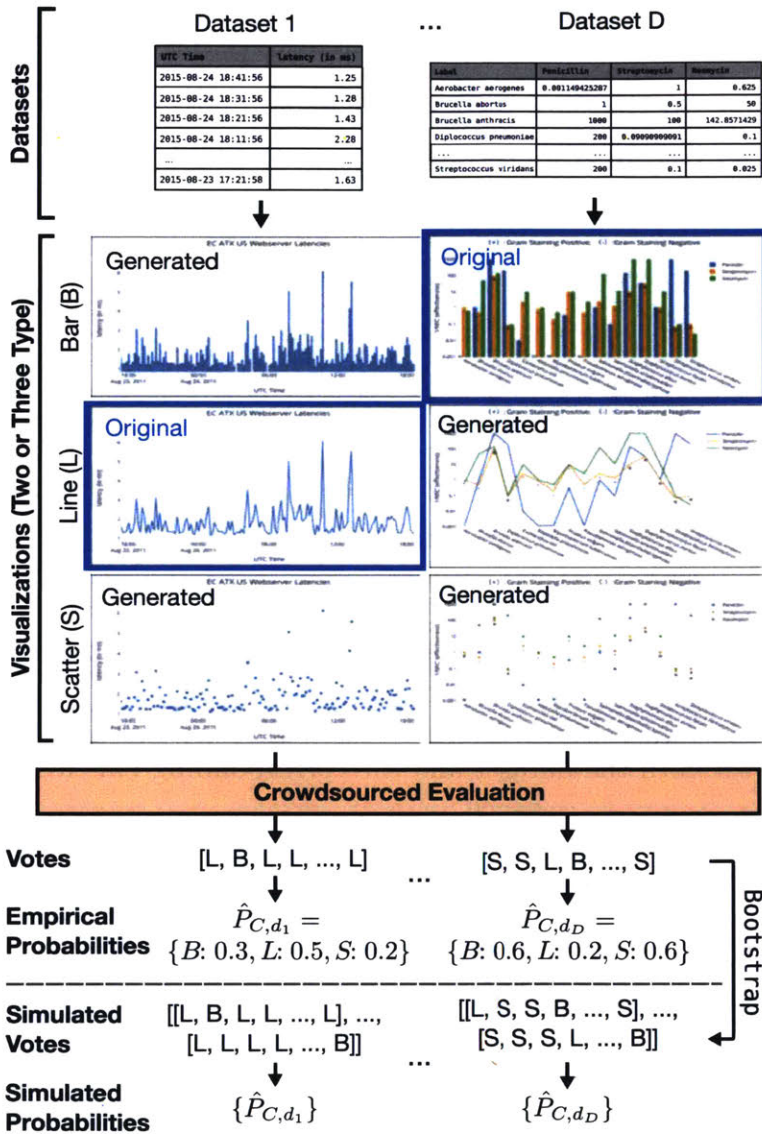


Figure 4-12: Experiment flow. The original user-generated visualizations are highlighted in blue, while we generated the visualizations of the remaining types. After crowdsourced evaluation, we have a set of votes for the best visualization type of that dataset. We calculate confidence intervals for model scores through bootstrapping.

feel you had enough knowledge to answer the previous questions?” Users could answer these questions on a six-point scale. Finally, users reported demographic information, educational attainment, occupation, technical background, and experience with data analysis or programming languages.

#### 4.6.1 Benchmarking Procedure

We use four types of predictors in our benchmark: human, rule-based model, ML-based model, and baseline. The two human predictors are the **Plotly** predictor, which is the visualization type of the original plot created by the Plotly user, and the **MTurk** predictor is the choice of a single random Mechanical Turk participant. When evaluating the performance of individual Mechanical Turkers, that individual’s vote was excluded from the set of votes used in the mode estimation.

The two rule-based predictors include one commercial system and another research system. The first, Tableau’s **Show Me** feature [106], is based on the expressiveness and effectiveness criteria of Mackinlay’s APT [105]. The second, the **CompassQL** recommender engine [71], powers the Voyager and Voyager 2 systems [199,200].

The two learning-based predictors are **DeepEye** and **Data2Vis**. In all cases, we tried to make choices that maximize prediction performance, within reason. We uploaded datasets to Show Me, DeepEye, and CompassQL as comma-separated values (CSV) files, and to Data2Vis as JSON objects. Unlike VizML and Data2Vis, DeepEye supports pie, bar, and scatter visualization types. We marked both pie and bar recommendations were both bar predictions, and scatter recommendations as line predictions in the two-type case.

For all tools, we modified the data within reason to maximize the number of valid results. For the remaining errors (4 for Data2Vis, 14 for DeepEye), and cases without

returned results (12 for DeepEye and 33 for CompassQL) we assigned a random chart prediction.

Predictor performance is evaluated as the total sum of normalized effectiveness scores. This *Consensus-Adjusted Recommendation Score* (CARS) of a predictor is defined as:

$$CARS_{predictor} = \frac{1}{|D|} \sum_{d \in D} \frac{\hat{P}_c(\hat{c}_{predictor, d})}{\max(\{\hat{P}_c\})} \times 100 \quad (4.6)$$

where  $|D|$  is the number of datasets (66 for two-class and 99 for three-class),  $\hat{c}_{predictor, d}$  is the predicted visualization type for dataset  $d$ , and  $\hat{P}_c$  returns the fraction of Mechanical Turker votes for a given visualization type. Note that the minimum CARS  $> 0\%$ . We establish 95% confidence intervals around these scores by comparing against  $10^5$  bootstrap samples of the votes, which can be thought of as synthetic votes drawn from the observed probability distribution.

#### 4.6.2 Benchmarking Results

We first measure the degree of consensus using the Gini coefficient, the distribution of which is shown in Figure 4-13. If a strong consensus was reached for all visualizations, then the Gini distributions would be strongly skewed towards the maximum, which is  $1/2$  for the two-class case, and  $2/3$  for the three-class case. Conversely, a lower Gini implies a weaker consensus, indicating an ambiguous ideal visualization type. The Gini distributions are not skewed towards either extreme, which supports the use of a soft scoring metric such as CARS over a hard measure like accuracy.

The Consensus-Adjusted Recommendation Scores for each model and task are visualized as a bar chart in Figure 4-14. We first compare the CARS of VizML ( $88.96 \pm 1.66$ ) against that of Mechanical Turkers ( $86.66 \pm 5.38$ ) and Plotly users ( $90.35 \pm 1.85$ ) for



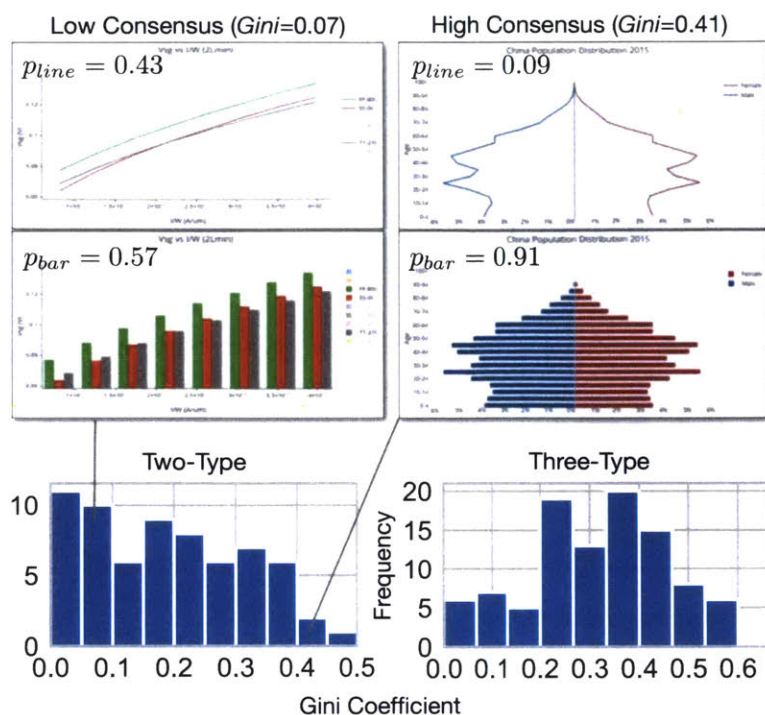
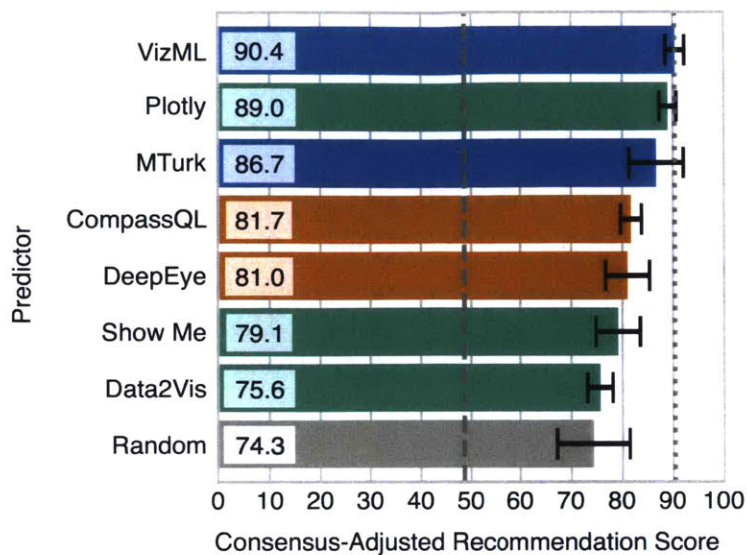


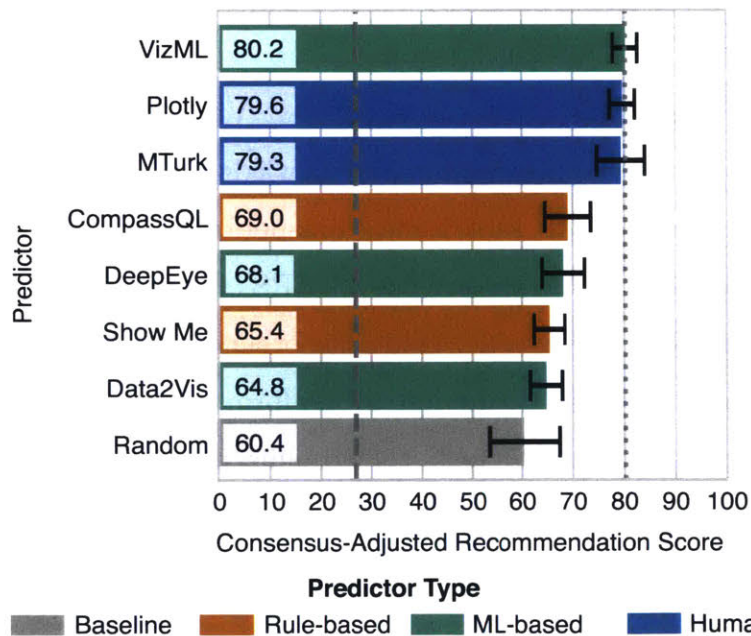
Figure 4-13: Distribution of Gini coefficients for the visualization type prediction tasks. Higher Gini indicates stronger consensus.

the two-class case, as shown in Figure 4-14a. It is surprising that VizML performs comparably to the original Plotly users, who possess domain knowledge and invested time into visualizing their own data. VizML significantly out-performs Data2Vis ( $75.61 \pm 2.44$ ) and DeepEye ( $79.12 \pm 4.33$ ). Show Me achieves a CARS of ( $81.70 \pm 2.05$ ), which is similar to that of CompassQL ( $80.98 \pm 4.32$ ). While the other recommenders were not trained to perform visualization type prediction, all perform slightly better than the random classifier ( $74.30 \pm 7.09$ ). For this task, the absolute minimum score was ( $48.61 \pm 2.95$ ).

The same results hold for the three-class case shown in Figure 4-14b, in which the CARS of VizML ( $81.18 \pm 2.39$ ) is slightly higher, but within error bars, than that of Mechanical Turkers ( $79.28 \pm 4.66$ ), and Plotly users ( $79.58 \pm 2.44$ ). Data2Vis ( $64.75 \pm 3.13$ ) and DeepEye ( $68.09 \pm 4.11$ ) outperform the Random ( $60.37 \pm 6.98$ ) with a larger margin, but still within error. CompassQL ( $68.95 \pm 4.48$ ) slightly surpasses Show Me ( $65.37 \pm 2.98$ ), also within error. The minimum score was ( $26.93 \pm 3.46$ ).



(a) Two-type (bar/line) visualization type CARS.



(b) Three-type (bar/line/scatter) visualization type CARS.

Figure 4-14: Consensus-Adjusted Recommendation Score of three ML-based, two rule-based, and two human predictors when predicting consensus visualization type. Error bars show 95% bootstrapped confidence intervals, with  $10^5$  bootstraps. The mean minimum achievable score is the lower dashed line, while the highest achieved CARS is the upper dotted line.

## 4.7 Discussion

In this chapter, we introduce VizML, a machine learning approach to visualization recommendation that uses a large corpus of datasets and corresponding visualizations. We identify five key prediction tasks and show that, relative to both random guessing and simpler classifiers, neural network classifiers attain high test accuracies on these tasks. We also benchmark with a test set established through crowdsourced consensus, and show that the performance of neural networks is comparable with the performance of individual humans.

Visualization system developers have multiple paths when it comes to incorporating ML-based recommenders such as VizML into authoring workflows. Partial specification recommenders on top of existing manual specification tools, such as the Show Me [106] feature in Tableau [173], rely on design choice suggestions that could be provided by a learned model. Code-based authoring environments such as the Draco [119] and Vega-Lite [153] editors could use partial specification recommenders to power visualization “autocomplete” features that would then suggest, in real time, design choices in response to user interaction. Mixed-initiative systems such as Voyager [200] and DIVE [66] could leverage Top-N recommendations to present a gallery of visualizations for users to search and drill-down. Indeed, designing interactions with ML-based recommenders is an important area of future work.

To develop ML-based recommenders for their own systems, developers could begin by identifying user design choices and extracting simple features from the data. Given sufficient volume, those features and design choices can be used to train models, as we have demonstrated in here. Alternatively, developers can overcome the cold-start problem by using pre-trained models such as VizML. With models in hand, developers can progress further by collecting the usage analytics (e.g., measures of engagement such as clicks and shares) to establish customized measures of visualization effective-

ness.

## 4.8 Future Work

We acknowledge the limitations of the Plotly corpus and our approach. First, despite aggressive deduplication, our model is certainly biased towards the Plotly dataset. As a web-based platform, Plotly could draw a certain cohort of analysts, encourage certain types of plots by interface design or defaults, or be more appropriate for specific types and sizes of data. Second, neither the Plotly user nor the Mechanical Turker is an expert in data visualization. Third, we acknowledge that VizML was only focused on a subset of the tasks usually considered in a visualization recommendation pipeline.

Promising avenues for future work lie in both data collection and modelling directions. On the data side, there is a need for more diverse training data from other tools (e.g., Many Eyes and Tableau) and for those pertaining to adjacent data science tasks such as feature selection and data transformation. Richer training data allows researchers to investigate the previous bias concerns, optimize visualization recommenders with a task-based (or, more generally, multi-objective) effectiveness metric, recommend multiple views of a dataset, study complementary approaches to feature engineering, and integrate distinct design choice recommendations using a probabilistic graphical model.

Despite the increasing prevalence of recommendation features within visualization tools, research progress in visualization recommendation has been impeded because of the lack of a standard benchmark. Without a benchmark, it is difficult to compare different approaches to this problem. Furthermore, it is difficult to bootstrap a recommender system without access to the results from a tool. Just as large repositories like ImageNet [35] and CIFAR-10 have played a significant role in shaping computer vision research, and serve as a useful benchmarking tool, the same should exist for

visualization recommendation.

## Chapter 5

### VizNet

*A Large-Scale Visualization Learning and Benchmarking Repository*

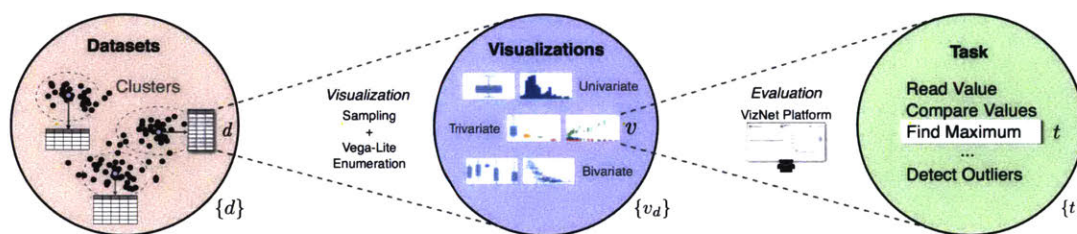


Figure 5-1: VizNet enables data scientists and visualization researchers to aggregate data, enumerate visual encodings, and crowdsource effectiveness evaluations.

A primary concern in visualization is how to effectively encode data values as visual variables. Beginning with Cleveland and McGill’s seminal work [30], researchers have studied this question of *graphical perception* by conducting experiments with human subjects. And increasingly, researchers are seeking to operationalize the guidelines these studies produce by using handcrafted rule-based systems [119, 198] or learned models [36, 64, 103], such as VizML, which was presented in CHAPTER 4.

To increase the scale and diversity of the subject pool, modern studies have eschewed traditional laboratory set-ups in favor of crowdsourcing platforms [58]. But a constraining factor for true ecological validity remains. Collecting, curating, and cleaning data is a laborious and expensive process and, thus, researchers have relied on running studies with ad hoc datasets. These datasets, sometimes synthetically generated, do not display the same characteristics as data found in the wild. Moreover, as one-off exemplars, their use makes it difficult to compare approaches against a common baseline.

Large-scale databases (such as WordNet [117] and ImageNet [35]) have proven to be instrumental in pushing the state-of-the-art forward because they provide the data

needed to train and test machine learning models, as well as a common baseline for evaluation, experimentation, and benchmarking. Their success has led researchers to call for a similar approach to advance data visualization [9, 44]. However, insufficient attention has been paid to design and engineer a centralized and large-scale repository for evaluating the effectiveness of visual designs.

In response, we introduce VizNet: a corpus of over 31 million datasets (657 GB of data) compiled from the web, open data repositories, and online visualization platforms. In characterizing these datasets, we find that they typically consist of 17 records describing 3 dimensions of data. 51% of the dimensions in the corpus record categorical data, 44% quantitative, and only 5% measure temporal information. Such high-level properties, and additional measures such as best statistical fit and entropy, contribute a taxonomy of real-world datasets that can inform assessments of ecological validity of prior studies.

This chapter demonstrates VizNet’s viability as a platform for conducting online crowdsourced experiments at scale by replicating the Kim and Heer (2018) study, which assessed the effect of task and data distribution on the effectiveness of visual encodings [78]. We extend this previous study by adding in an additional task: outlier detection. While largely in line with the original findings, our results do exhibit several statistically significant differences as a result of our more diverse backing datasets. These differences inform our discussion on how crowdsourced graphical perception studies must adapt to and account for the variation found in organic datasets. VizNet, along with data collection and analysis scripts, is publicly available at <https://viznet.media.mit.edu>.

Data visualization is an inherently combinatorial design problem: a single dataset can be visualized in a multitude of ways, and a single visualization can be suitable for a range of analytic tasks. As the VizNet corpus grows, assessing the effectiveness of these (*data, visualization, task*) triplets, even when using crowdsourcing, will

quickly become time- and cost-prohibitive. To contend with this scale, we conclude by formulating effectiveness prediction as a machine learning task over these triplets. We demonstrate a proof-of-concept model that predicts the effectiveness of unseen triplets with non-random performance. Our results suggest that machine learning offers a promising method for efficiently annotating VizNet content. VizNet provides an important opportunity to advance our understanding of graphical perception.

## 5.1 Data

VizNet incorporates four large-scale corpora, assembled from the web, online visualization tools, and open data portals.

### 5.1.1 Corpora

The first category of corpora includes data tables harvested from the web. In particular, we use horizontal relational tables from the WebTables 2015 corpus [19], which extracts structured tables from the Common Crawl. In these tables, entities are represented in rows and attributes in columns.

The second type of corpus includes tabular data uploaded by users of two popular online data visualization and analysis systems. Plotly [130] is a software company that develops visualization tools and libraries. Once created, Plotly charts can be posted to the Plotly Community Feed [132]. Using the Plotly API, we collected approximately 2.5 years of public visualizations from the feed, starting from 2015-07-17 and ending at 2018-01-06. The second system, ManyEyes [188] allowed users to create and publish visualizations through a web interface. It was available from 2007–2015, and was used by tens of thousands of users [120].

The third type of corpus includes public data from the Open Data Portal Watch [118,



122], which catalogs and monitors 262 open data portals such as `data.noaa.gov` from CKAN, `finances.worldbank.org` from Socrata, and `opendata.brussels.be` from OpenDataSoft. The majority of these portals are hosted by governments, and collect civic and social data.

VizNet aggregates these corpora into a centralized repository. However, the majority of datasets are from WebTables. Therefore, in the following sections, we describe each corpus individually with 250K randomly sampled datasets, to avoid oversampling the WebTable corpus. We combine these datasets into a balanced sample of one million datasets, which we refer to as the **VizNet 1M corpus**.

### 5.1.2 Characterization

Summary statistics and underlying distributions of each of the five corpora are shown in Figure 5-2. The data type of a column is classified as either categorical, quantitative, or temporal, which we abbreviate as C, Q and T, respectively. This data type is detected using a heuristic-based approach that incorporates column name and value information. For quantitative columns, we use the Kolmogorov-Smirnov test [110] to examine the goodness-of-fit of six distributions: the normal, log-normal, exponential, power law, uniform and chi-squared distributions. We reject the null hypothesis of a distribution fit if the p-value of the associated test is lower than the level  $\alpha = 0.05$ . If all distributions are rejected at  $\alpha$ , we consider the distribution to be undefined. If multiple distributions are not rejected, we consider the “best” fit to be that with the highest p-value. We also report the skewness and percent of outliers, defined as data points that fall more than  $1.5 \times IQR$  below the first quartile or above the third quartile, where  $IQR$  is the interquartile range. The statistical distribution of categorical columns within each corpus is characterized using the normalized entropy.

Corpus	Source	Size		Dimensions		Types			Statistical	
		# Data	Gb	Cols	Rows	C (%)	Q (%)	T (%)	Distribution	Entropy
WebTables 2015	Web	90.26M	137	4	5	57.58	35.56	6.86	norm, log-norm, power	0.94
Plotly	Tool	1M	140	3	50	17.29	75.47	7.24	log-norm, norm, power	0.68
Many Eyes	Tool	311K	14	2	19	51.58	46.04	2.48	norm, log-norm, expon	0.81
ODPW	Repository	269K	366	2	70	76.55	21.20	2.24	norm, log-norm, power	0.50
VizNet 1M	Combined	1M	405	3	17	50.71	44.58	4.71	norm, log-norm, power	0.79

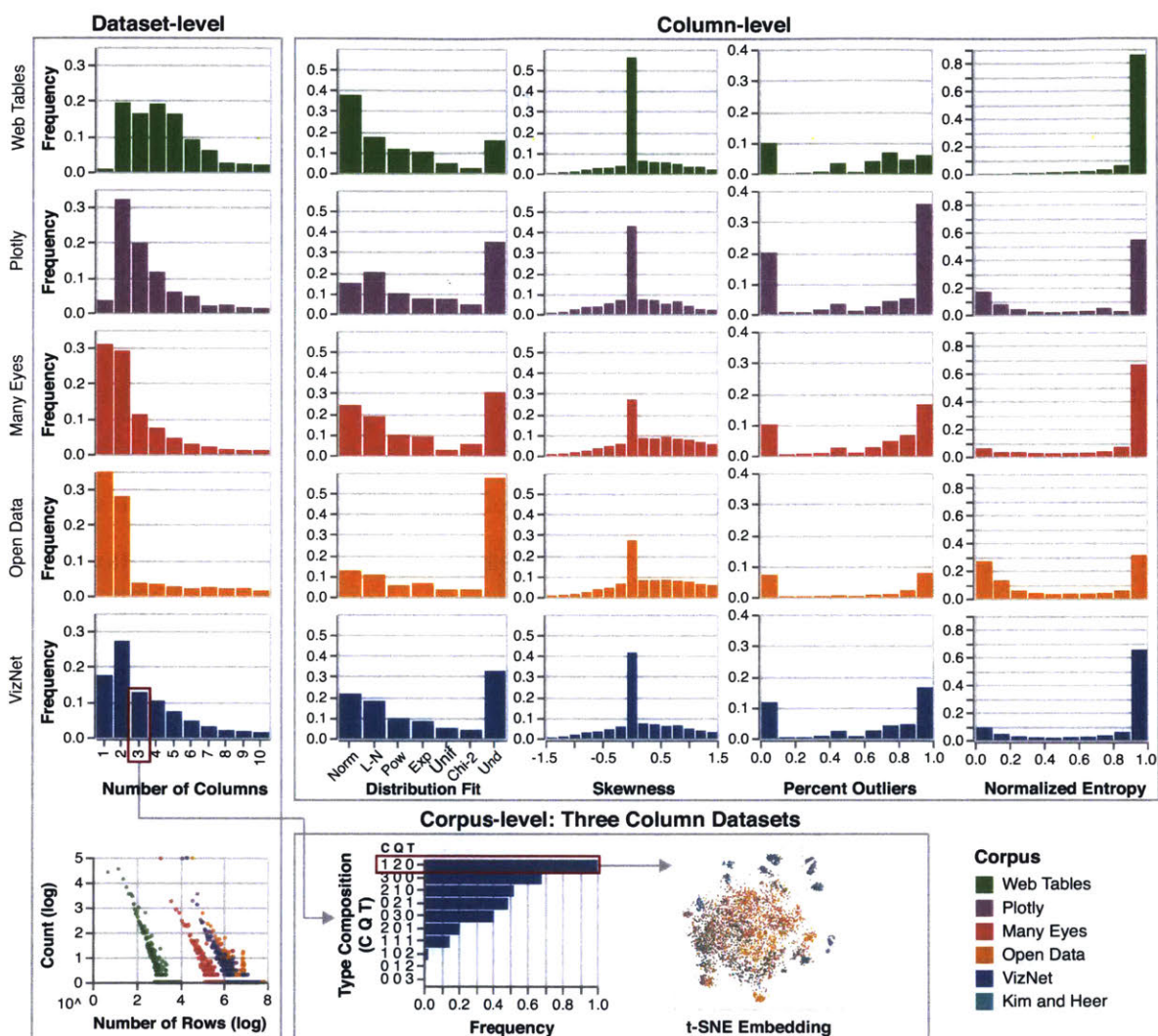


Figure 5-2: Summary statistics (top) and distributions (bottom) of the four source corpora and the VizNet 1M corpus. In the top table, we report the median number of rows and columns. The Distribution column includes the top three most frequent column distributions. Distributions are abbreviated as Norm = normal, L-N = log-normal, Pow = power law, Exp = exponential, Unif = uniform, and Und = undefined. The bottom part of the figure contains distributions describing columns, datasets, and the entire corpus. The bars outlined in red represent three column datasets and the subset which contain one categorical and two quantitative fields. The clustering of three column (C=1, Q=2) datasets is shown in more detail in Figure 5-5.

## 5.2 Experiment Design

To evaluate the utility of VizNet as a resource for data scientists and visualization researchers, we conducted an experiment where we first replicated the Kim and Heer (2018) prior study [78] using real-world datasets from the VizNet corpus to assess the influence of user task and data distribution on visual encoding effectiveness. These datasets were sampled to match constraints from the prior study and ensure that participants only saw valid data. We then extended this experiment by including an additional task on outlier detection. Finally, we trained a machine learning model that learns the perceptual effectiveness of different visual designs and evaluated its predictive power across test datasets.

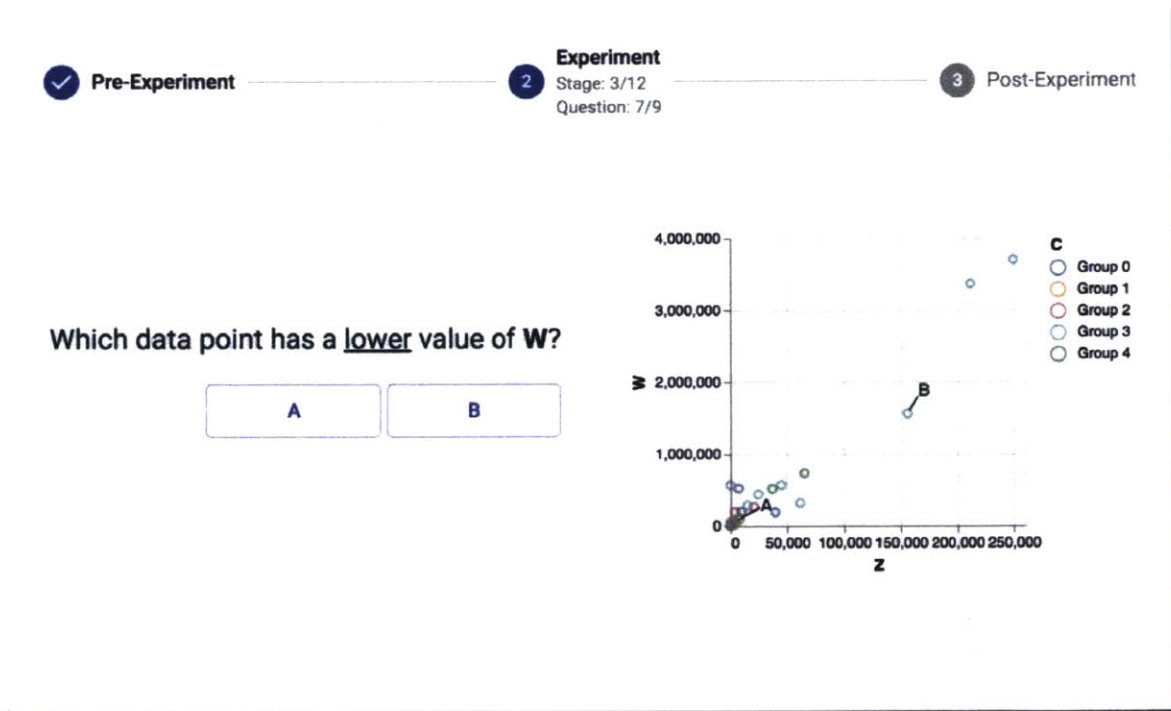


Figure 5-3: VizNet user interface for the *Compare Values* task experiment.

### 5.2.1 Replication of Kim and Heer (2018)

Kim and Heer (2018), “*Assessing Effects of Task and Data Distribution on the Effectiveness of Visual Encodings*,” conducted a crowdsourced experiment measuring subject performance (*i.e.* error rate and response time) across data distributions ( $\mathbf{D}$ ), visualization designs ( $\mathbf{V}$ ), and task types ( $\mathbf{T}$ ). The 24 data distributions characterize trivariate data involving one categorical and two quantitative fields ( $C=1$ ,  $Q=2$ ) sampled from 2016 daily weather measurements [113] according to univariate entropies of the quantitative fields, cardinalities, and number of records per category.

The authors employed a mixed design using a within-subjects treatment for visual encodings and between-subjects treatments for tasks and data characteristics. They analyzed responses from 1,920 participants on Amazon’s Mechanical Turk (MTurk), who individually completed 96 questions and 12 engagement checks, and calculated the absolute and ranked performance of different ( $\mathbf{D} \times \mathbf{V} \times \mathbf{T}$ ) conditions, as well as the interaction effects between different data characteristics, visual channels, and task types. These results extended existing models of encoding effectiveness, such as APT [105], and provided valuable insights for automatic visualization design systems.

### 5.2.2 Datasets

For this experiment, we sampled VizNet datasets according to a procedure that matched constraints from Kim and Heer (2018) and ensured that participants only saw valid data without missing values. This procedure was developed after an initial pilot study with a subset of the corpus in which all datasets were manually verified.

To begin, we identified all datasets with more than one categorical field and two quantitative fields ( $C \geq 1$  and  $Q \geq 2$ ). Then, we sampled all possible three column subsets with exactly one categorical and two quantitative fields ( $C=1$ ,  $Q=2$ ). Following this

sampling, we filtered out datasets using a number of constraints. First, we rejected datasets containing any null values. Second, we required that the column names of all datasets must contain between 1 and 50 ASCII-encoded characters. Third, we limited the cardinality (*e.g.* the number of unique groups) of the categorical columns between 3 and 30. Fourth, we restricted the group names between 3 and 30 characters, at least one of which is alphanumeric. Lastly, we required that each of the groups must contain 3 to 30 values. We chose these values to be consistent with the upper and lower constraints of Kim and Heer (2018).

Our sampling procedure resulted in 2,941 valid datasets from the Open Data Corpus (100,626 possible combinations), 6,090 valid datasets from Many Eyes (354,206 combinations), 1,368 from Plotly (347,387 combinations), and 82,150 from a subset of the Webtables corpus (1,512,966 combinations). From this set of candidates, we randomly selected 200 candidates per visualization specification  $\times$  task condition. We use  $\mathbf{V}$  to denote the number of visualization specifications and  $\mathbf{T}$  to denote the number of tasks, which leads to 60 such conditions ( $\mathbf{V} \times \mathbf{T} = 12 \times 5 = 60$ ). The 200 number of datasets sampled from the VizNet corpus is consistent with the 192 datasets sampled in Kim and Heer (2018). As a result, this sampling resulted in  $200 \times 12 = 2,400$  datasets per task, 2,400 datasets per corpus, and  $9,600 = 2,400 \times 4$  total datasets.

### 5.2.3 Visual Encodings

We selected the twelve visual encoding specifications chosen in Kim and Heer (2018). These encodings are specified using the Vega-Lite grammar [153], which specifies plots using a geometric mark type (*e.g.* bar, line, point) and a mapping from data fields to visual encoding channels (*e.g.*  $x$ ,  $y$ , *color*, *shape*, and *size*). In particular, Kim and Heer (2018) used twelve visualization designs, all of which are scatterplots (a *point*

mark) with different mappings between data and encoding channels.

We used the *Tableau-10* scheme for color encoding categorical fields with cardinality less than 10, and *Tableau-20* for categorical fields with cardinality greater than or equal to 20. For positional encodings, in contrast to Kim and Heer (2018), we used a heuristic to determine whether an axis should start at zero. If the range of a variable  $Q$  is less than 10% of maximum value  $0.1 \times |\max(Q)|$ , then we default to Vega-lite axis ranges. Based on a pilot study, we found that this heuristic was necessary to ensure that no questions were prohibitively difficult.

#### 5.2.4 Tasks

Following Kim and Heer (2018), we considered 4 visualization tasks informed by the Amar et al. (2005) [6] taxonomy of low-level analytic activities. Two of those tasks were *value tasks*: *Read Value* and *Compare Values* asked users to read and compare individual values. The other two tasks were *summary tasks*: *Find Maximum* and *Compare Averages* required the identification or comparison of aggregate properties. Each of these tasks was formulated as a binary question (two-alternative forced choice questions). We generated the two alternatives using the procedure described in the prior study.

#### 5.2.5 Procedure

Identical to Kim and Heer (2018), we also employed a mixed design incorporating a within-subjects treatment for visual encodings and a between-subjects treatment for tasks. Each participant answered 9 questions (1 attention check and 8 real) for each of the 12 visual encodings, presented in a random order. Every participant was assigned to a specific task. Unlike Kim and Heer (2018), we did not incorporate dataset

conditions. Each dataset was selected randomly from the pool of 200 datasets per  $\mathbf{V} \times \mathbf{T}$  condition. In order to ensure reliable human judgment, we followed the process from Kim and Heer (2018) and incorporated 12 evenly distributed gold standard tasks. The gold standard tasks presented a user with a real dataset encoded in the present visual encoding condition, and asked what information is presented in the visual channel that encodes the first quantitative column ( $Q_1$ ).

### 5.2.6 Participants

Crowdsourcing platforms such as MTurk are widely used to recruit participants and conduct online experiments at scale [79, 109]. We recruited in total 1,342 MTurk workers who were located in the U.S. and had  $\geq 95\%$  HIT approval rating.

During the analysis, we included the following criteria to ensure the quality of human judgment: we selected subjects who accurately answered 100% of the gold standard questions, had an experimental error rate of less than 60%, and can effectively distinguish colors. We had set the gold standard response exclusion threshold to 100% (i.e., discarding responses if even 1 out of these 12 questions was answered incorrectly). We have verified that a more lenient 80% exclusion threshold does not significantly change the results. Kim and Heer (2018) does not report a dropout rate, making it difficult to assess whether and by how much our dropout rate differs. We included two Ishihara color blindness plate tests [69] along with two pre-screen questions to ensure the participants can effectively distinguish colors. A total of 96.47% reported no vision deficiency and were allowed to participate in the experiment. This resulted in a total of 624 participants' data for in the analysis.

Of the 624 participants, 43.75% were male, 55.44% female, and 0.48% non-binary. 6.38% of the participants had no degree, whereas others had bachelor's (43.10%), master's (14.90%), Ph.D. (3.04%), associate (14.58%) degrees as well as a high school

diploma (17.46%). Each participant received 1.00 USD in compensation, which we calculated using the average times of a pilot study and the same hourly wage of Kim and Heer (2018).

### 5.3 Results

In this section, we describe the results of our experiment, compare them with the results of Kim and Heer (2018) [78], and demonstrate a machine learning-based approach to predicting effectiveness from (*data, visualization, task*) triplets.

#### 5.3.1 Comparing Subject Performance

We first compared subject performance with the quantitative results of Kim and Heer (2018) by considering aggregate error rates and log response times per visualization specification and task condition ( $\mathbf{V} \times \mathbf{T} = 12 \times 4$ ). Following this, we calculated mean error rates with 95% bootstrapped confidence intervals, performed by sampling participants with replacement. To analyze the difference of mean error rates and response times we conducted permutation tests with  $10^4$  permutations. We test significance at a significance level of  $\alpha = 0.05$  with Bonferroni correction for our  $m = 48$  hypotheses. The results for the error rate and log response times are shown in Figure 5-4.

The absolute error rates of our replication tend to agree with those of Kim and Heer (2018) for the *Read Value* task, and to a lesser extent for the *Compare Values* task. The rankings of different visual encodings are also similar. However, for the the *summary tasks* (*Find Maximum* and *Compare Averages*), our observed error rates depart from those of Kim and Heer (2018). Though more data points are needed to draw meaningful conclusions, these results suggest that real-world data affects error rates for more complex tasks.



In contrast, the absolute response times in our study seem to be systematically longer for all tasks except the *Compare Values* task. However, the relative rankings of different encoding are consistent with those of Kim and Heer (2018).

### 5.3.2 Extending with an Outlier Detection Task

As suggested by Kim and Heer (2018), investigating additional task types is a promising direction of future research. In particular, tasks with more subjective definitions, such as *Cluster* and *Find Anomalies* were not included in Kim and Heer (2018). Nevertheless, as outlier detection is one of the most important data analysis tasks in practice, it warrants further empirical study. We extended the prior work by considering this latter task of identifying “which data cases in a set  $S$  of data cases have unexpected/exceptional values.”

We generated 2,400 datasets using the sampling methodology described in the previous section. First, we presented users with a definition of outliers as “observations that lie outside the overall pattern of distribution.” Then, using the same experiment design, we assessed answers to the question “Are there outliers in  $Q_1$ ?” “Yes” and “No” are provided as response options. Outliers were determined using the median absolute deviation (MAD)-based approach described in [93], which is robust to varying sample sizes, compared to other simple approaches.

We found that the error rates for the outlier detection task are higher compared to the other tasks (see Figure 5-4). This may be due to an inadequate measure of ground truth, inconsistent definitions, or lack of prior training. It is important to note that the specification rankings resemble that of the *Read Value* task: *color* and *size* trail behind other encodings channels. Conversely, the log response times are significantly shorter than for other tasks, for all except the faceted charts with *row* encodings.

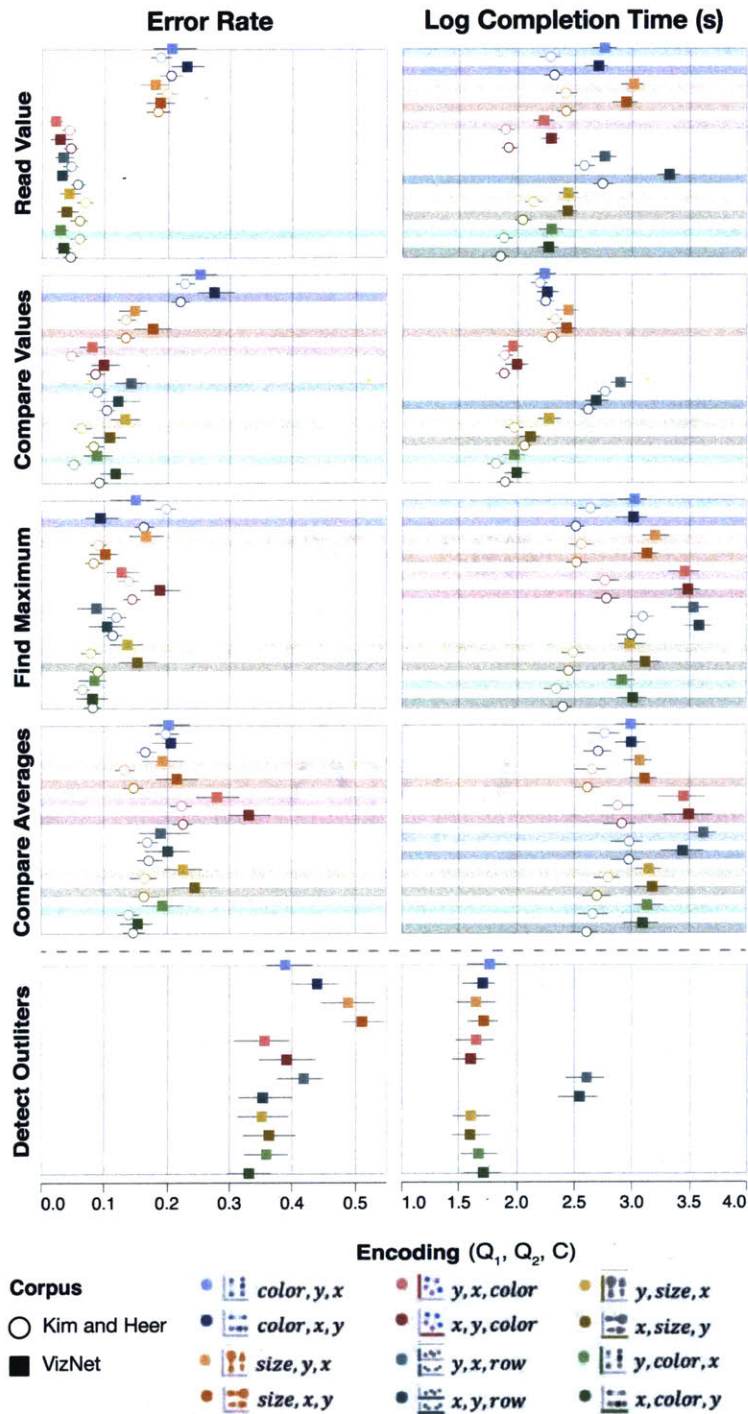


Figure 5-4: Bootstrapped means and 95% confidence intervals for error rates (left) and log response times (right) across tasks and visual encodings for Kim and Heer (2018) original data, and our replication on VizNet. We reuse the original color encoding of Kim and Heer (2018). Shading indicates a statistically significant difference.

### 5.3.3 Learning a Model to Predict Effectiveness

To characterize a dataset, we extracted 167 features: 60 per quantitative field Q, 11 for the categorical field C, 15 for the Q-Q pair, 6 for the two C-Q pairs, and 9 which consider all three fields. These features characterized summary statistics (*e.g.* coefficient of variance and kurtosis), statistical distributions (*e.g.* entropy and statistical fits), pairwise relationships (*e.g.* correlations and one-way ANOVA p-values), clusteredness and spatial autocorrelation.

We first decoded diversity within our space of datasets using these features. Using principal components analysis, we computed 32 principal components which collectively explain over 85% of the variance within our dataset. Then, we generated a two-dimensional t-SNE projection of these principal components, as shown in Figure 5-5. It is important to note that the datasets used in Kim and Heer (2018) [78] are highly clustered and separate from the datasets used within our replication. This observation is robust for different numbers of principal components and values of perplexity (5-200).

To predict log completion time we use gradient boosted regression trees, a model with strong “off-the-shelf” performance. Training on 80% sample of the data, we were able to predict log completion times in a 20% hold-out test set with a 5-fold cross-validated  $R^2$  of 0.47, which strongly outperforms baseline models such as K-nearest neighbors and simple linear regression. A scatter plot of observed vs. predicted values for the top performing model is shown in Figure 5-6. Learning curves in Figure 5-7 indicate that, despite the large number of features, our model does not overfit on the training set, and that there are still gains from increasing the number of training samples.

Kim and Heer (2018) reports the trade-off between response time and error rate. To capture this trade-off, we created a combined metric from the log response times

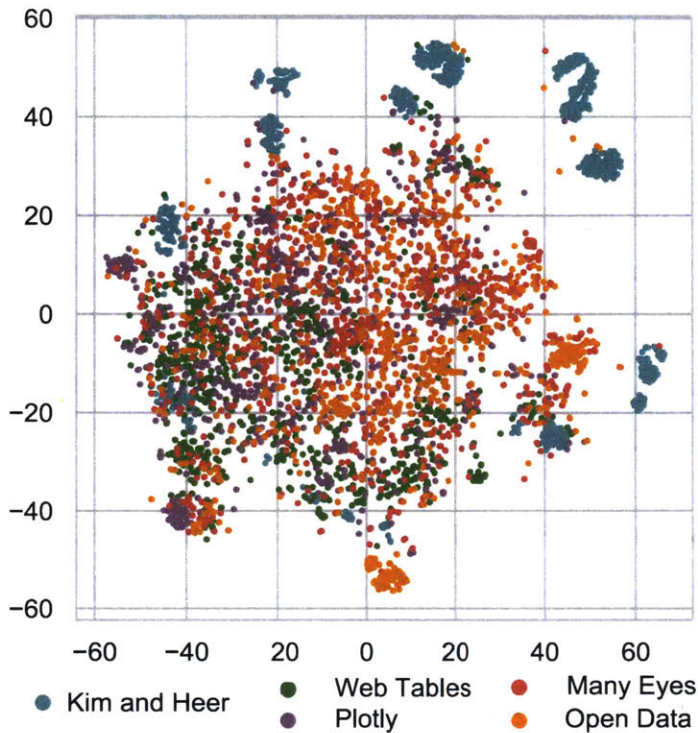


Figure 5-5: Two-dimensional t-SNE projection of datasets with one categorical and two quantitative columns, evenly sampled from Kim and Heer (2018) and the four corpora within VizNet, with a perplexity of 75.

and error rate metrics by partitioning the log response times into 20% quantiles, and the error rates into five bins of equal width, for a total of 25 pairs. Then, we characterized each  $(d, v, t)$  triplet with the associated (response time + error rate) pair, and resampled minority classes using the Synthetic Minority Over-sampling Technique (SMOTE) [25]. Training a gradient boosted classification tree on the balanced training set resulted in a Top-3 prediction accuracy of 52.48%.

### 5.3.4 Limitations

Although we have successfully demonstrated the effectiveness of VizNet, it is important to acknowledge limitations. Replication and reproducibility are essential to advance research [126]. In the experiment, we attempted to replicate Kim and Heer (2018) as closely as possible. However, due to practical constraints, we introduced clarifying modifications to the question text and interface design. Due to variance

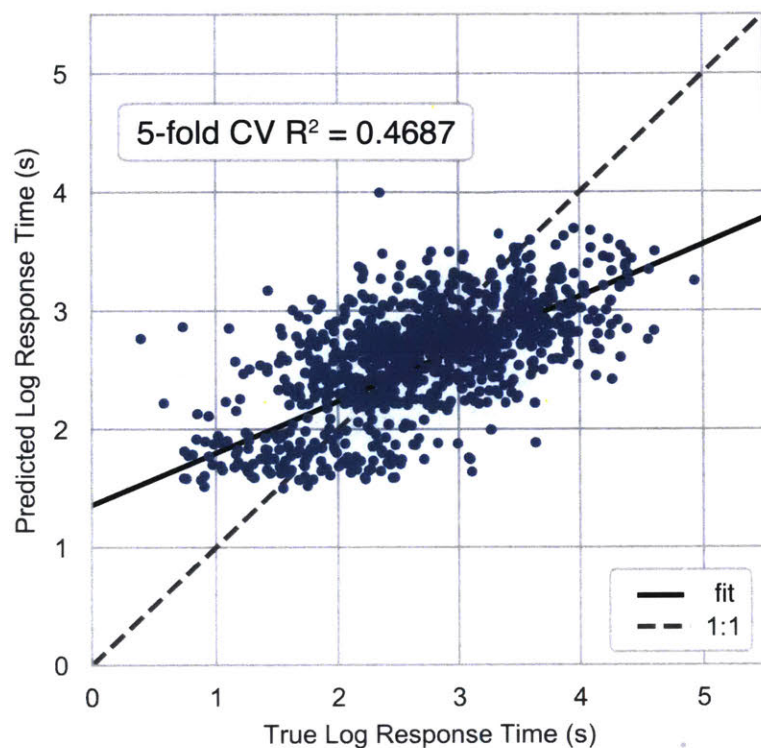


Figure 5-6: Observed log response times (in seconds) vs. those predicted by a gradient boosted regression tree. The dotted diagonal line denotes a perfect relationship between observation and prediction.

between crowd workers, we were not able to recruit the same participants; nor do we control for question difficulty, which is calibrated in Kim and Heer (2018). Most of all, we did not exactly replicate the original conditions of the synthetic datasets, which would have limited the amount of real-world VizNet datasets available for sampling. Notwithstanding these limitations, our work provides an important direction to understand the opportunities and challenges faced in replicating prior work in human-computer interaction and visualization research.

With respect to extending the experiment to include an additional task, we note that outlier detection, unlike the other tasks, does not have a defined ground truth. Though we used a robust outlier detection method, there may be a limitation to any purely quantitative method that does not rely on human consensus. The lack of an objective notion of outliers and absence of a clear definition thereof in the questions, reinforces the inconsistency between ground truth and crowdsourced labels presumably partially explaining the consistently high error rate. In the context of the

machine learning model, while human judgments can play an important role in help predicting perceptual effectiveness, crowdsourced training data can be noisy. The current experiment was unable to analyze lower bound requirements of quality data, but VizNet’s diverse dataset offers such opportunity for future research.

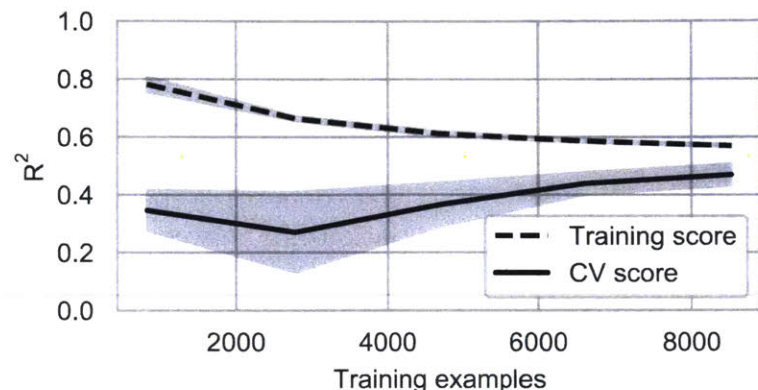


Figure 5-7: Training  $R^2$  and 5-fold cross-validation  $R^2$  as the number of training examples increases.

#### 5.4 Discussion

There are several important areas where VizNet makes important contributions. VizNet provides a noteworthy contribution to advance our knowledge of effective graphical perception by enabling scientific community access to rich datasets for visualization learning, experimentation, replication, and benchmarking. VizNet offers both the full corpus and the sampled corpus of one million datasets (VizNet 1M). It further described the dimensions, types, and statistical properties of these datasets. The voluminous collection of VizNet complements synthetically generated data. Moreover, the properties of the VizNet corpus can inform assessments of the ecological validity of other corpora from domains beyond VizNet.

#### **Implications of enabling the VizNet interface for the scientific community.**

We envision that in the long run, adoption of a common corpus and benchmarks by the visualization community will facilitate the sharing and comparing of results at scale. We have made VizNet publicly available at <https://viznet.media.mit.edu>. A taxonomy in VizNet is formed by splitting our corpus first on the number of columns

of a datasets, and then on the composition of column types. Therefore, we should design interactions to help users query, filter, sample datasets within this taxonomy (*e.g.* give me all datasets with one categorical, two quantitative, and one temporal field). Moreover, this informs the need for supporting keyword search to allow filtering by domain, in addition to filtering on other dataset properties (*e.g.* give me highly correlated datasets with exactly two quantitative fields).

**Implications of VizNet for replication and experimentation.** We replicate Kim and Heer (2018) to demonstrate the utility of using VizNet. Our results with real-world data are largely consistent with their findings. As a result of our more diverse backing datasets, however, there are statistically significant differences in error rates for the complex tasks. We also note that task completion times with real data are consistently longer for all but one task. These discrepancies suggest that graphical perception studies must account for the variation found in real datasets. Kim and Heer (2018) acknowledge this direction of future work by describing the need for investigating “all [data] distributions of potential interest.” The process of harvesting these diverse distributions would be facilitated by using VizNet. We further extend the original experiment by considering an additional “detect outliers” task, an important but subjective visual analysis task that is difficult to assess using synthetic data.

**Implications of VizNet for learning a metric of perceptual effectiveness.** While Kim and Heer (2018) employed a mixed effects model to analyze their results, we proposed to conceive the harvested data as a collection of (*data, visualization, task*) triplets, each of which is associated with effectiveness measures. Using machine learning models, we predicted the completion time with an  $R^2$  value of 0.47. Acknowledging the trade-off between completion time and error rate, we constructed a combined metric and achieved a top-3 prediction accuracy of 52.48%. Despite the noise and skew of crowdsourced labels, and a relatively small sample size, these results

out-perform both random chance and baseline classifiers. In doing so, they illustrate the potential for learning a metric of perceptual effectiveness from experimental results.

## 5.5 Summary and Future Work

Large-scale data collection efforts for facilitating research are common across sciences and engineering, from genomics to machine learning. Their success in accelerating the impact of research in their respective fields is a testament to the importance of easy access to large-scale realistic data, as well as benchmarking and performing research on shared databases. As the field of data visualization research grows from its infancy, we expect the need for and utility of large-scale data and visualization repositories to significantly grow as well. VizNet is a step forward in addressing this need.

We plan to extend VizNet along three major directions: (1) incorporate and characterize more datasets, (2) harness the wisdom of the crowd, and (3) develop active learning algorithms for optimal experiment design. These extensions aim to increase the utility of VizNet not for visualization research but for data-related research more broadly.

**Incorporate and characterize more datasets.** VizNet currently centralizes four corpora of data from the web, open data portals, and online visualization galleries. We plan to expand the VizNet corpus with the 410,554 Microsoft Excel workbook files (1,181,530 sheets) [26] that were extracted from the ClueWeb09 web crawl<sup>1</sup>. Furthermore, Morton et al. [120] report that 73,000 Tableau workbooks and 107,500 datasets from Tableau Public, all of which could be integrated into VizNet. Lastly, we plan to incorporate 10,663 datasets from Kaggle<sup>2</sup>, of which 1,161 datasets are included alongside the R statistical environment<sup>3</sup>, and to leverage the Google Dataset

---

<sup>1</sup><http://lemurproject.org/clueweb09.php>

<sup>2</sup><https://www.kaggle.com/datasets>

<sup>3</sup><https://github.com/vincentarelbundock/Rdatasets>



Search<sup>4</sup> to source more open datasets.

In the future work, we plan to characterize the semantic content within column and group names by using natural language processing techniques such as language detection, named entity recognition, and word embeddings. Moreover, as we describe the features of the datasets within the VizNet corpus, we can characterize the bias between corpora in terms of dimensions, type composition, and statistical properties of columns. This will enable us to systematically study the extent to which these corpora differ. The existence of such bias between corpora can be seen in Section 4.2. A clearer understanding of between-corpus bias could inform future techniques for sampling from the VizNet corpus.

**Harness the wisdom of the crowd.** Domain-specific crowdsourcing platforms such as FoldIt, EteRNA, GalaxyZoo, and Game with Purpose, have incentivized citizen scientists to discover new forms of proteins [32], RNAs [91], galaxies [97], and artificial intelligence algorithms [189]. We envision VizNet will enable citizen scientists and visualization researchers to execute graphical perception experiments at scale. In recent years, crowdsourcing has been pivotal in the creation of large-scale machine learning corpora. Daemo [48], a self-governed crowdsourcing marketplace, was instrumental in the creation of the Stanford Question Answering Dataset (SQuAD) [139], whereas MTurk was used to curate the ImageNet dataset [35].

The effectiveness of crowdsourcing has also been exemplified in our experiment as we collected human judgments for the evaluation of visual designs. It is interesting to note that some of the crowd workers enjoyed the intellectual aspect of the experiment, as illustrated by their post-experiment responses: (1) *‘I found this survey entertaining, it makes you think and use your head’* (2) *‘It is a very interesting survey to carry out since it promotes the capacity of analysis I congratulate you for that’*. A natural progression to harness crowdsourcing mechanisms for VizNet includes an extension

---

<sup>4</sup><https://toolbox.google.com/datasetsearch>

of the literature on task design [80], crowd work quality improvements [41, 89], and incentive design [49, 189].

**Develop active learning for optimal experiment design.** Although gathering human-judgment labels for each triplet is costly, it is possible to learn the effectiveness from these labeled triplets to label unseen ones (see Section §4.4). To further illustrate this strategy, we conducted a small experiment on the same data as in Section §4.4, where the completion times are categorized into low, medium, and high. To propagate labels, we employed self-learning [3], so we added the model predictions that have a high certainty to the labeled set. The predictions with a low certainty were replaced with crowdsourced labels following the uncertainty algorithm [31]. Figure 5-8 shows how this strategy improves the accuracy on a test set after a number of iterations against the baseline of training on all of the labeled samples (supervised learning). In the future, we plan to harness active learning to assess the quality of human judgment.

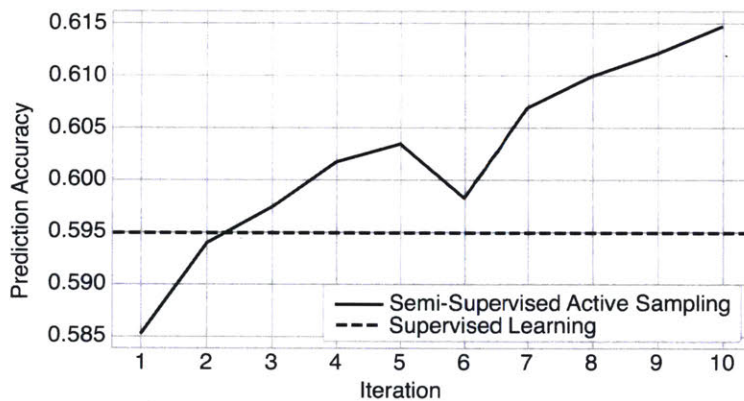


Figure 5-8: Performance curves obtained by semi-supervised active learning and supervised learning over 10 iterations.

**Demonstrate utility for data systems research.** The creation of VizNet was motivated by the need for ecologically valid benchmarks within visualization research. However, a repository of real-world tabular datasets is potentially valuable for the broader community of data systems researchers. The data science workflow spans many tasks, ranging from data collection, to preparation, to exploration, to presentation. Each task could benefit from an ecologically valid benchmark and, with a proper training signal, automation through machine learning.

## Chapter 6

# Sherlock

### *A Machine Learning Approach to Semantic Type Detection*

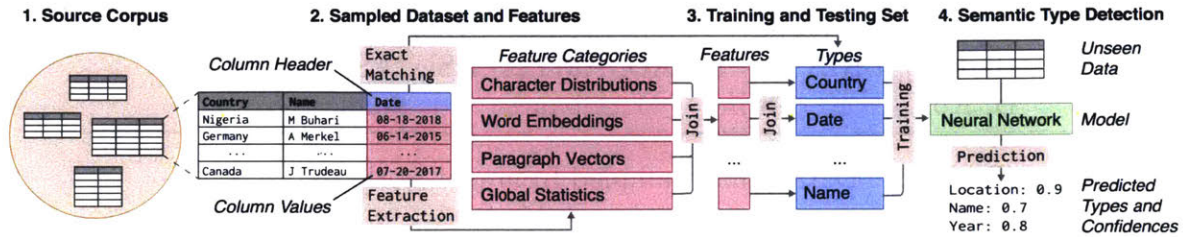


Figure 6-1: Data processing and analysis flow starting from (1) a corpus of real-world datasets, proceeding to (2) feature extraction, (3) mapping from the features to ground truth semantic types from column headers, and then moving to (4) model training and prediction.

Data preparation and analysis systems rely on correctly detecting the types of data columns to enable and constrain functionality. For example, visualization recommender systems such as DIVE and VizML rely on type-dependent rules. For example, automated data cleaning facilitates the generation of clean data through validation and transformation rules that depend on the data type [73, 140]. Schema matching identifies the correspondences between data objects, and frequently uses data types to constrain the search space of correspondences [138, 206]. Data discovery surfaces data relevant to a given query, often relying on semantic similarities across tables and columns [22, 23].

While most systems reliably detect *atomic types* such as `string`, `integer`, and `boolean`, *semantic types* are disproportionately more powerful, and in many cases, essential. Semantic types provide finer-grained descriptions of the data by establishing correspondences between the columns and real-world concepts and, as such, they can help with schema matching to determine which columns refer to the same real-world concepts, or with data cleaning by determining the conceptual domain of

Type	Sampled values
location	TBA — Chicago, Ill. — Detroit, Mich. — Nashville, Tenn.
location	UNIVERSITY SUITES — U.S. 27; NA — NORSE HALL
location	Away — Away — Home — Away — Away
date	27 Dec 1811 — 1852 — 1855 — - — 1848 — 1871 — 1877
date	--, 1922 — --, 1902 — --, 1913 — --, 1919
date	December 06 — August 23 — None
name	Svenack — Svendd — Sveneldritch — Svengöran
name	HOUSE, BRIAN — HSIAO, AMY — HSU, ASTRID
name	D. Korb — K. Moring — J. Albanese — l. dunn





Table 6.1: Data values sampled from real-world datasets.

a column. In some cases, the detection of a semantic types can be easy. For example, an `ISBN` or `credit card number` are generated according to strict validation rules, lending themselves to straightforward type detection with just a few rules. But most types, including `location`, `birth date`, and `name`, do not adhere to such a structure, as shown in Table 6.1.

Existing open source and commercial systems take *matching-based* approaches to semantic type detection. For example, regular expression matching captures the patterns of data values by using predefined character sequences. Dictionary approaches use matches between data headers and values with internal look-up tables. While sufficient for detecting simple types, these matching-based approaches are often not robust to malformed or dirty data, support only a limited number of types, and under-perform for types without strict validations. For example, Figure 6-2 shows that Tableau detects a column labeled “Continent Name” as `string`. After removing the column headers, no semantic types are detected. Note that missing headers or incomprehensible headers are not uncommon. For example, SAP’s system table `T005` contains country information and column `NMFMT` is the standard name field, whereas `INTCA` refers to the ISO code or `XPLZS` to zip-code.

Machine learning models, coupled with large-scale training and benchmarking cor-

### Detected Types With Column Headers

Country/Region	String	Latitude	Longitude	Country/Region	String
	Abc				Abc
country-capitals.csv	country-capitals.csv	country-capit...	country-capital...	country-capitals.csv	country-capitals.csv
Country Name	Capital Name	Latitude	Longitude	Country Code	Continent Name
Aruba	Oranjestad	12.517	-70.033	AW	North America
Australia	Canberra	-35.267	149.133	AU	Australia
Austria	Vienna	48.200	16.367	AT	Europe

### Detected Types Without Column Headers

String	String	Decimal	Decimal	String	String
Abc	Abc	#	#	Abc	Abc
country-capitals-edite...	country-capitals-edi...	country-capit...	country-capital...	country-capitals-edite...	country-capitals-edited....
F1	F2	F3	F4	F5	F6

Remove Headers

Figure 6-2: Data types detected by Tableau Desktop 2018.3 for a dataset of country capitals, with and without headers.

pora, have proven effective at predictive tasks across domains. Examples include the AlexNet neural network trained on ImageNet for visual recognition and the Google Neural Machine Translation system pre-trained on WMT parallel corpora for language translation. Inspired by these advances, we introduce **Sherlock**, a deep learning approach to semantic type detection trained on a large corpus of real-world columns.

To begin, we consider 78 semantic types described by T2Dv2 Gold Standard,<sup>1</sup> which matches the properties from the DBpedia ontology with column headers from the WebTables corpus. Then, we use exact matching between the semantic types and column headers to extract 686,765 data columns from the VizNet corpus presented in CHAPTER 5, a large-scale repository of real-world datasets collected from the web, popular visualization systems, and open data portals.

We consider each column as a mapping from the column values to a column header.

<sup>1</sup><http://webdatacommons.org/webtables/goldstandardV2.html>

We then extract 1,588 features from each column, describing the distribution of characters, semantic content of the words and columns, and global statistics such as cardinality and uniqueness. Treating the column headers as ground truth labels of the semantic type, we formulate semantic type detection as a multiclass classification problem.

A multi-input neural network architecture achieves a support-weighted  $F_1$ -score of 0.89, exceeding that of a decision tree baseline model, two matching-based approaches, and the consensus of the crowdsourced annotations. We then examine the types for which the neural network demonstrates high and low performance, investigate the contribution of each feature category to model performance, extract feature importances from the decision tree baseline, and present an error-reject curve that indicates the potential of combining learned models with human annotations.

To conclude, we discuss promising avenues for future research in semantic type detection, such as assessing training data quality at scale, enriching feature extraction processes, and establishing shared benchmarks. To support benchmarks for future research, we open source our data, code, and trained model.<sup>2</sup> For developers wishing to integrate Sherlock into existing systems, we distribute a pretrained model as a Python library.<sup>3</sup>

## 6.1 Related Work

Sherlock is informed by existing *commercial and open source systems* for data preparation and analysis, as well as prior research work on *ontology-based*, *feature-based*, *probabilistic*, and *synthesized* approaches to semantic type detection.

---

<sup>2</sup>**Code and data:** <https://sherlock.media.mit.edu/data>

<sup>3</sup>**Python library:** <https://sherlock.media.mit.edu/library>

**Commercial and open source** Semantic type detection enhances the functionality of commercial data preparation and analysis systems such as Microsoft Power BI [115], Trifacta [180], and Google Data Studio [53]. To the best of our knowledge, these commercial tools rely on manually defined regular expression patterns dictionary lookups of column headers and values to detect a limited set of semantic types. For instance, Trifacta detects around 10 types (e.g., **gender** and **zip code**) and Power BI only supports time-related semantic types (e.g., **date/time** and **duration**). Open source libraries such as messytables [87], datalib [185], and csvkit [54] similarly use heuristics to detect a limited set of types. Benchmarking directly against these systems was unfeasible due to the small number of supported types and lack of extensibility. However, we compare against learned regular expression and dictionary-based benchmarks representative of the approaches taken by these systems.

**Ontology-based** Prior research work, with roots in the semantic web and schema matching literature, provide alternative approaches to semantic type detection. One body of work leverages existing data on the web, such as WebTables [19], and ontologies (or, knowledge bases) such as DBPedia [7], Wikitology [175], and Freebase [12]. Venetis et al. [186] construct a database of value-type mappings, then assign types using a maximum likelihood estimator based on column values. Syed et al. [175] use column headers and values to build a Wikitology query, the result of which maps columns to types. Informed by these approaches, we looked towards existing ontologies to derive the 275 semantic types considered in this paper.

**Feature-based** Several approaches capture and compare properties of data in a way that is ontology-agnostic. Ramnandan et al. [141] use heuristics to first separate numerical and textual types, then describe those types using the Kolmogorov-Smirnov (K-S) test and Term Frequency-Inverse Document Frequency (TF-IDF), respectively. Pham et al. [129] use slightly more features, including the Mann-Whitney test for

numerical data and Jaccard similarity for textual data, to train logistic regression and random forest models. We extend these feature-based approaches with a significantly larger set of features that includes character-level distributions, word embeddings, and paragraph vectors. Orders of magnitude more features and training samples motivates our use of high-capacity machine learning models such as neural networks. While code for benchmarking these models was not available, we include a decision tree model to represent “simpler” machine learning models.

**Probabilistic** The third category of prior work employs a probabilistic approach. Goel et al. [51] use conditional random fields to predict the semantic type of each value within a column, then combine these predictions into a prediction for the whole column. Limaye et al. [94] use probabilistic graphical models to annotate values with entities, columns with types, and column pairs with relationships. These predictions simultaneously maximize a potential function using a message passing algorithm. Probabilistic approaches are complementary to our machine learning-based approach by providing a means for combining column-specific predictions. However, as with prior feature-based models, code for retraining these models was not made available for benchmarking.

**Synthesized** Puranik [136] proposes a “specialist approach” combining the predictions of regular expressions, dictionaries, and machine learning models. More recently, Yan and He [204] developed a system that, given a search keyword and set of positive examples, synthesizes type detection logic from open source GitHub repositories. This system provides a novel approach to leveraging domain-specific heuristics for parsing, validating, and transforming semantic data types. While both approaches are exciting, the code underlying these systems was not available for benchmarking.



## 6.2 Data

We describe the semantic types we consider, how we extracted data columns from a large repository of real-world datasets, and our feature extraction procedure.

### 6.2.1 Data Collection

Ontologies such as WordNet [182] and DBpedia [7] describe semantic concepts, properties of such concepts, and relationships between them. To constrain the number of types we consider, we adopt the types described by the T2Dv2 Gold Standard,<sup>1</sup> the result of a study matching DBpedia properties [147] with columns from the Web Tables web crawl corpus [19]. These 275 DBpedia properties, such as **country**, **language**, and **industry**, represent semantic types commonly found in datasets scattered throughout the web.

To expedite the collection of real-world data from diverse sources, we use the VizNet repository [65], which aggregates and characterizes data from two popular online visualization platforms and open data portals, in addition to the Web Tables corpus. For feasibility, we restricted ourselves to the first 10M Web Tables datasets, but considered the remainder of the repository. We then match data columns from VizNet that have headers corresponding to our 275 types. To accommodate variation in casing and formatting, single word types matched case-altered modifications (e.g., **name** = **Name** = **NAME**) and multi-word types included concatenations of constituent words (e.g., **release date** = **releaseDate**).

The matching process resulted in 6,146,940 columns matching the 275 considered types. Manual verification indicated that the majority of columns were plausibly described by the corresponding semantic type, as shown in Table 6.1. In other words, matching column headers as ground truth labels of the semantic type yielded high

quality training data.

### 6.2.2 Feature Extraction

To create fixed-length representations of variable-length columns, aid interpretation of results, and provide “hints” to our neural network, we extract features from each column. To capture different properties of columns, we extract four categories of features: global statistics (27), aggregated character distributions (960), pretrained word embeddings (200), and self-trained paragraph vectors (400).

**Global statistics** The first category of features describes high-level statistical characteristics of columns. For example, the “column entropy” feature describes how uniformly values are distributed. Such a feature helps differentiate between types that contain more repeated values, such as `gender`, from types that contain many unique values, such as `name`. Other types, like `weight` and `sales`, may consist of many numerical characters, which is captured by the “mean of the number of numerical characters in values.” A complete list of these 27 features can be found in Table 6.2.

**Character-level distributions** Preliminary analysis indicated that simple statistical features such as the “fraction of values with numerical characters” provide surprising predictive power. Motivated by these results and the prevalence of character-based matching approaches such as regular expressions, we extract features describing the distribution of characters in a column. Specifically, we compute the count of all 96 ASCII-printable characters (i.e., digits, letters, and punctuation characters, but not whitespace) within each value of a column. We then aggregate these counts with 10 statistical functions (i.e., any, all, mean, variance, min, max, median, sum, kurtosis, skewness), resulting in 960 features. Example features include “whether all values

Feature description
Number of values.
Column entropy.
Fraction of values with unique content.*
Fraction of values with numerical characters.*
Fraction of values with alphabetical characters.
Mean and std. of the number of numerical characters in values.*
Mean and std. of the number of alphabetical characters in values.*
Mean and std. of the number special characters in values.*
Mean and std. of the number of words in values.*
{Percentage, count, only/has-Boolean} of the None values.
{Stats, sum, min, max, median, mode, kurtosis, skewness, any/all-Boolean} of length of values.

Table 6.2: Description of the 27 global statistical features. Asterisks (\*) denote features included in Venetis et al. [186].

contain a ‘-’ character” and the “mean number of ‘/’ characters.”

**Word embeddings** For certain semantic types, columns frequently contain commonly occurring words. For example, the `city` type contains values such as *New York City*, *Paris*, and *London*. To characterize the semantic content of these values, we used word embeddings that map words to high-dimensional fixed-length numeric vectors. In particular, we used a pre-trained GloVe dictionary [127] containing 50-dimensional representations of 400K English words aggregated from 6B tokens, used for tasks such as text similarity [76]. For each value in a column, if the value is a single word, we look up the word embedding from the GloVe dictionary. We omit the a term if it did not appear in the GloVe dictionary. For values containing multiple words, we looked up each distinct word and represented the value with the mean of the distinct word vectors. Then, we computed the mean, mode, median and variance of word vectors across all values in a column.

**Paragraph vectors** To represent entire columns with numerical vectors, we implemented the Distributed Bag of Words version of Paragraph Vectors (DBoW-PV) [90]. Paragraph vectors were originally developed to learn representations for pieces of texts, but have proven effective for more general tasks, such as document similarity [33]. In our implementation, each column is a “paragraph” while values within a column are “words”: both the entire column and constituent values are represented by one-hot encoded vectors. We then randomly select a value vector, concatenate the remaining column and value vectors, and train a model to predict the former from the latter. Using the Gensim library [145] and a holdout set, we trained this model for 20 iterations. We used this model to map columns to 400-dimensional paragraph vectors, which provided a balance between predictive power and computational tractability.

### 6.2.3 Filtering and Preprocessing

Certain types occur more frequently in the VizNet corpus than others. For example, **description** and **city** are more common than **collection** and **continent**. To address this heterogeneity, we limited the number of columns to at most 15K per class and excluded the 10% types containing less than 1K columns.

Other semantic types, especially those describing numerical concepts, are unlikely to be represented by word embeddings. To contend with this issue, we filtered out the types for which at least 15% of the columns did not contain a single word that is present in the GloVe dictionary. This filter resulted in a final total of **686,765 columns** corresponding to **78 semantic types**, of which a list is included in Table 6.3. The distribution of number of columns per semantic type is shown in Figure 6-3.

Before modeling, we preprocess our features by creating an additional binary feature indicating whether word embeddings were successfully extracted for a given column.

Semantic Types				
Address	Code	Education	Notes	Requirement
Affiliate	Collection	Elevation	Operator	Result
Affiliation	Command	Family	Order	Sales
Age	Company	File size	Organisation	Service
Album	Component	Format	Origin	Sex
Area	Continent	Gender	Owner	Species
Artist	Country	Genre	Person	State
Birth date	County	Grades	Plays	Status
Birth place	Creator	Industry	Position	Symbol
Brand	Credit	ISBN	Product	Team
Capacity	Currency	Jockey	Publisher	Team name
Category	Day	Language	Range	Type
City	Depth	Location	Rank	Weight
Class	Description	Manufacturer	Ranking	Year
Classification	Director	Name	Region	
Club	Duration	Nationality	Religion	

Table 6.3: 78 semantic types included in this study.

Including this feature results in a total of **1,588 features**. Then, we impute missing values across all features with the mean of the respective feature.

## 6.3 Methods

We describe our deep learning model, decision tree baseline, two matching-based benchmarks, and crowdsourced consensus benchmark. Then, we explain our training and evaluation procedures.

### 6.3.1 Sherlock: A Multi-input Neural Network

Prior machine learning approaches to semantic type detection [94,186] trained simple models, such as logistic regression, on relatively small feature sets. We consider a significantly larger number of features and samples, which motivates our use of a feedforward neural network. Specifically, given the different number of features and

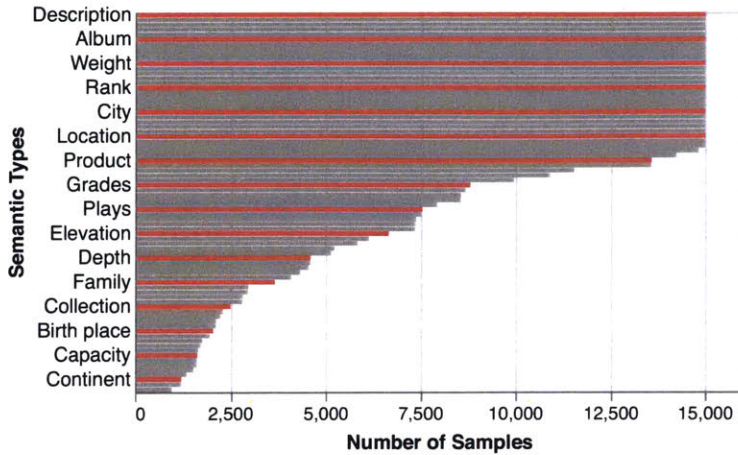


Figure 6-3: Number of columns per semantic type extracted from VizNet after filtering out the types with less than 1K columns and more than 15% of columns not present in the GloVe dictionary.

varying noise levels within each feature category, we use a multi-input architecture with hyperparameters shown in Figure 6-4.

At a high-level, we train subnetworks for each feature category except the statistical features, which consist of only 27 features. These subnetworks “compress” input features to an output of fixed dimension. We chose this dimension to be equal to the number of types in order to evaluate each subnetwork independently. Then, we concatenate the weights of the three output layers with the statistical features to form the input layer of the primary network.

Each network consists of two hidden layers with rectified linear unit (ReLU) activation functions. Iteration over hidden layer sizes between 100 and 1,000 (i.e., on the order of the input layer dimension) resulted in hidden layer sizes of 300, 200, and 400 for the character-level, word embedding, and paragraph vector subnetworks, respectively. To prevent overfitting, we included drop out layers and weight decay terms. The neural network, which we refer to as “Sherlock,” is implemented in TensorFlow [1].

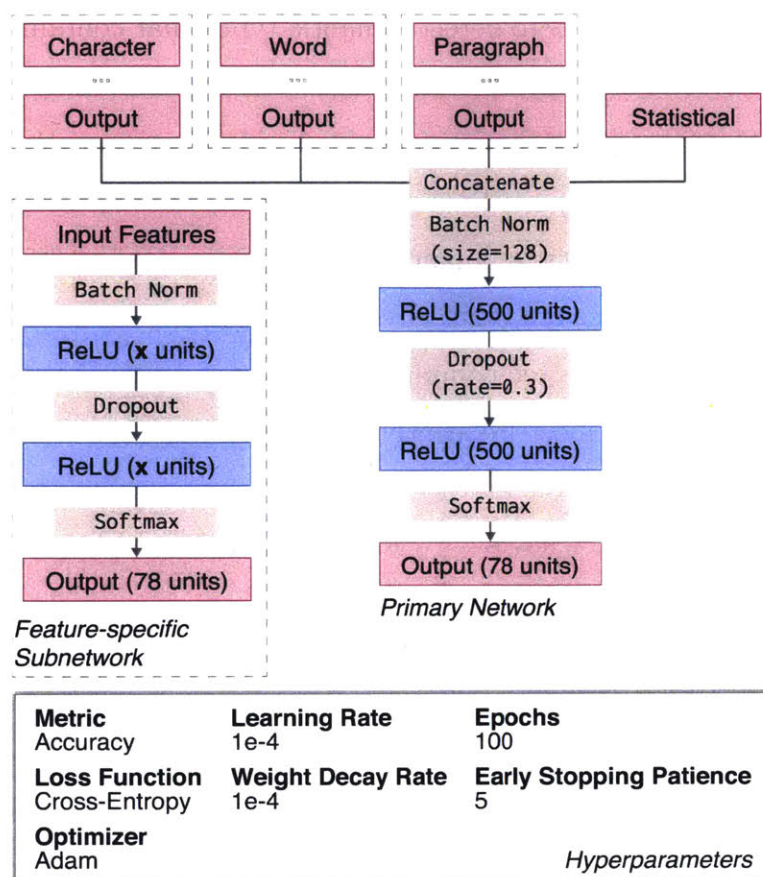


Figure 6-4: Architecture and hyperparameters of feature-specific sub-networks and primary network.

### 6.3.2 Benchmarks

To measure the relative performance of Sherlock, we compare against four benchmarks.

**Decision tree** The first benchmark is a decision tree, a non-parametric machine learning model with reasonable “out-of-the-box” performance and straightforward interpretation. We use the decision tree to represent the simpler models found in prior research, such as the logistic regression and random forest models used in Pham et al. [129]. Learning curves indicated that decision tree performance plateaued beyond a depth of 50, which we then used as the maximum depth. For the remaining parameters, we used the default settings in the scikit-learn package [125].

**Dictionary** Dictionaries are commonly used to detect semantic types that contain a finite set of valid values, such as `country`, `day`, and `language`. The first matching-based benchmark is a dictionary that maps column values or headers to semantic types. For each type, we collected the 1,000 most frequently occurring values across all columns, resulting in 78,000 { `value` : `type` } pairs. For example, Figure 6-5 shows examples of entries mapped to the `grades` type. Given an unseen data column at test time, we compare 1,000 randomly selected column values to each entry of the dictionary, then classify the column as the most frequently matched type.

<b>Dictionary Entries</b> (20 out of 1000)				
9-12	KG - 05	06 - 08	KG - 12	KG - 08
K-5	PRESCHOOL-5	PK -	K-8	- 12
PK - 05	6-8	PRESCHOOL-8	06 - 12	PK - 12
09 - 12	KG-06	PK - 8	K-^	PK - 08

**Learned Regular Expression**  
`\w\w \-(?: \w\w)*+[06PK][A-Za-z]*+\-\w\w\w\w\w\w\w\w \w\w \w\w\w \w\w`

Figure 6-5: Examples of dictionary entries and a learned regular expression for the `grades` type.

**Learned regular expressions** Regular expressions are frequently used to detect semantic types with common character patterns, such as `address`, `birth date`, and `year`. The second matching-based benchmark uses patterns of characters specified by learned regular expressions. We learn regular expressions for each type using the evolutionary procedure of Bartoli et al. [8]. Consistent with the original setup, we randomly sampled 50 “positive values” from each type, and 50 “negative” values from other types. An example of a learned regular expression in Java format for the `grades` type is shown in Figure 6-5. As with the dictionary benchmark, we match 1,000 randomly selected values against learned regular expressions, then use majority vote to determine the final predicted type.

**Crowdsourced annotations** To assess the performance of human annotators at predicting semantic type, we conducted a crowdsourced experiment. The experiment began by defining the concepts of data and semantic type, then screened out partic-



ipants unable to select a specified semantic type. After the prescreen, participants completed three sets of ten questions separated by two attention checks. Each question presented a list of data values, asked “Which one of the following types best describes these data values?”, and required participants to select a single type from a scrolling menu with 78 types. Questions were populated from a pool of 780 samples containing 10 randomly selected values from all 78 types.

We used the Mechanical Turk crowdsourcing platform [79] to recruit 390 participants that were native English speakers and had  $\geq 95\%$  HIT approval rating, ensuring high-quality annotations. Participants completed the experiment in 16 minutes and 22 seconds on average and were compensated 2 USD, a rate slightly exceeding the United States federal minimum wage of 7.25 USD.

Of the 390 participants, 57.18% were male and 0.43% female. 1.5% completed some high school without attaining a diploma, while others had associates (10.5%), bachelor’s (61.0%), master’s (13.1%), or doctorate or professional degree (1.8%) in addition to a high school diploma (12.3%). 26.4% of participants worked with data daily, 33.1% weekly, 17.2% monthly, and 11.0% annually, while 12.3% never work with data. In terms of age: 10.0% of participants were between 18-23, 24-34 (60.3%), 35-40 (13.3%), 41-54 (12.6%), and above 55 (3.8%).

Overall, 390 participants annotated 30 samples each, resulting in a total of 11,700 annotations, or an average of 15 annotations per sample. For each sample, we used the most frequent (i.e., the mode) type from the 15 annotations as the crowdsourced consensus annotation.

### 6.3.3 Training and Evaluation

To ensure consistent evaluation across benchmarks, we divided the data into 60/20/20 training/validation/testing splits. To account for class imbalances, we evaluate model performance using the average  $F_1$ -score =  $2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$ , weighted by the number of columns per class in the test set (i.e., the support). To estimate the mean and 95% percentile error of the crowdsourced consensus  $F_1$  score, we conducted  $10^5$  bootstrap simulations by resampling annotations for each sample with replacement.

Computational effort and space required at prediction time are also important metrics for models incorporated into user-facing systems. We measure the average time in seconds needed to extract features and generate a prediction for a single sample, and report the space required by the models in megabytes.

## 6.4 Results

We report the performance of our multi-input neural network and compare against benchmarks. Then, we examine types for which Sherlock demonstrated high and low performance, the contribution of each feature category in isolation, decision tree feature importances, and the effect of rejection threshold on performance.

### 6.4.1 Benchmark Results

We compare Sherlock against decision tree, dictionary-based, learned regular expression, and crowdsourced consensus benchmarks. Table 6.4 presents the  $F_1$  score weighted by support, runtime in seconds per sample, and size in megabytes of each model.

Method	F <sub>1</sub> Score	Runtime (s)	Size (Mb)
<i>Machine Learning</i>			
Sherlock	0.89	0.42 ( $\pm 0.01$ )	6.2
Decision tree	0.76	0.26 ( $\pm 0.01$ )	59.1
<i>Matching-based</i>			
Dictionary	0.16	0.01 ( $\pm 0.03$ )	0.5
Regular expression	0.04	0.01 ( $\pm 0.03$ )	0.01
<i>Crowdsourced Annotations</i>			
Consensus	0.32 ( $\pm 0.02$ )	33.74 ( $\pm 0.86$ )	–

Table 6.4: Support-weighted F<sub>1</sub> score, runtime at prediction, and size of Sherlock and four benchmarks.

We first note that both machine learning models significantly outperform the matching-based and crowdsourced consensus benchmarks, in terms of F<sub>1</sub> score. Inspection of the matching-based benchmarks suggests that dictionaries and learned regular expressions are prone to “overfitting” on the training set. Feedback from crowdsourced workers suggests that annotating semantic types with a large number of types is a challenging and ambiguous task.

Comparing the two machine learning models, Sherlock significantly outperforms the decision tree baseline. However, the decision tree still demonstrates strong performance relative to other benchmarks. For cases in which interpretability of features and predictions are important considerations, decision trees may be a suitable choice of model.

Despite poor predictive performance, matching-based benchmarks are significantly smaller and faster than both machine learning models. For cases in which absolute runtime and model size are critical, optimizing matching-based models may be a worthwhile approach. This trade-off also suggests a hybrid approach of combining matching-based models for “easy” types with machine learning models for more ambiguous types.

### 6.4.2 Performance for Individual Types

Table 6.5 displays the top and bottom five types, as measured by the  $F_1$  score achieved by Sherlock for that type. High performing types such as `grades` and `industry` frequently contain a finite set of valid values, as shown in Figure 6-5 for `grades`. Other types such as `birth date` and `ISBN`, often follow consistent character patterns, as shown in Table 6.1.

Type	$F_1$ Score	Precision	Recall	Support
<i>Top 5 Types</i>				
Grades	0.991	0.989	0.994	1765
ISBN	0.986	0.981	0.992	1430
Birth Date	0.970	0.965	0.975	479
Industry	0.968	0.947	0.989	2958
Affiliation	0.961	0.966	0.956	1768
<i>Bottom 5 Types</i>				
Brand	0.685	0.760	0.623	574
Person	0.630	0.654	0.608	579
Director	0.537	0.700	0.436	225
Sales	0.514	0.568	0.469	322
Ranking	0.468	0.612	0.349	439

Table 6.5: Top five and bottom five types by  $F_1$  score.

Examples	True type	Predicted type
<i>Low Precision</i>		
81, 13, 3, 1	Rank	Sales
316, 481, 426, 1, 223	Plays	Sales
\$, \$\$, \$\$\$, \$\$\$\$, \$\$\$\$\$	Symbol	Sales
<i>Low Recall</i>		
#1, #2, #3, #4, #5, #6	Ranking	Rank
3, 6, 21, 34, 29, 36, 54	Ranking	Plays
1st, 2nd, 3rd, 4th, 5th	Ranking	Position

Table 6.7: Examples of low precision and low recall types.

To understand types for which Sherlock performs poorly, we include incorrectly pre-

dicted examples for the lowest precision type (**sales**) and the lowest recall type (**ranking**) in Table 6.7. From the three examples incorrectly predicted as **sales**, we observe that purely numerical values or values appearing in multiple classes (e.g., currency symbols) present a challenge to type detection systems. From the three examples of incorrectly predicted **ranking** columns, we again note the ambiguity of numerical values.

### 6.4.3 Contribution by Feature Category

We trained feature-specific subnetworks in isolation and report the  $F_1$  scores in Table 6.9. Word embedding, character distribution, and paragraph vector feature sets demonstrate roughly equal performance to each other, and significantly above that of the global statistics features, though this may be due to fewer features. Each feature set in isolation performs significantly worse than the full model, supporting our combining of each feature set.

Feature set	Num. Features	$F_1$ Score
Word embeddings	200	0.79
Character distributions	960	0.78
Paragraph vectors	400	0.73
Global statistics	27	0.25

Table 6.9: Performance contribution of isolated feature sets.

### 6.4.4 Feature Importances

We measure feature importance by the total reduction of the Gini impurity criterion brought by that feature to the decision tree model. The top 10 most important features from the global statistics and character-level distributions sets are shown in Table 6.10. While word embedding and paragraph vector features are important, they are difficult to interpret and are therefore omitted.

(a) Top-10 global statistics features (out of 27).

Rank	Feature Name	Score
1	Number of Values	1.00
2	Maximum Value Length	0.79
3	Mean # Alphabetic Characters in Cells	0.43
4	Fraction of Cells with Numeric Characters	0.38
5	Column Entropy	0.35
6	Fraction of Cells with Alphabetical Characters	0.33
7	Number of None Values	0.33
8	Mean Length of Values	0.28
9	Proportion of Unique Values	0.22
10	Mean # of Numeric Characters in Cells	0.16

(b) Top-10 character-level distribution features (out of 960).

Rank	Feature Name	Score
1	Sum of 'D' across values	1.00
2	Mean number of 'M'	0.77
3	Minimum number of '-'	0.69
4	Skewness of ','	0.59
5	Whether all values have a ','	0.47
6	Maximum number of 'g'	0.45
7	Skewness of ']'	0.45
8	Mean number of ','	0.40
9	Mean number of 'z'	0.37
10	Sum of 'n'	0.36

Table 6.10: Top-10 features for the decision tree model. “Score” denotes normalized gini impurity.

Inspecting Table 6.11a, we find that the “number of values” in a column is the most important feature. Certain classes like `name` and `requirements` tended to contain fewer values, while others like `year` and `family` contained significantly more values. The second most important feature is the “maximum value length” in characters, which may differentiate classes with long values, such as `address` and `description`, from classes with short values, such as `gender` and `year`.

The top character-level distribution features in Table 6.11b suggest the importance of

specific characters for differentiating between types. The third most important feature, the “minimum number of ‘-’ characters”, likely helps determine datetime-related types. The fifth most important feature, “whether all values have a ‘,’ character” may also distinguish datetime-related or name-related types. Further study of feature importances for semantic type detection is a promising direction for future research.

### 6.4.5 Rejection Curves

Given unseen data values, Sherlock assesses the probability of those values belonging to each type, then predicts the type with the highest probability. Interpreting probabilities as a measure of confidence, we may want to only label samples with high confidence of belonging to a type. To understand the effect of confidence threshold on predictive performance, we present the error-rejection curves of Sherlock and the decision tree model in Figure 6-6.

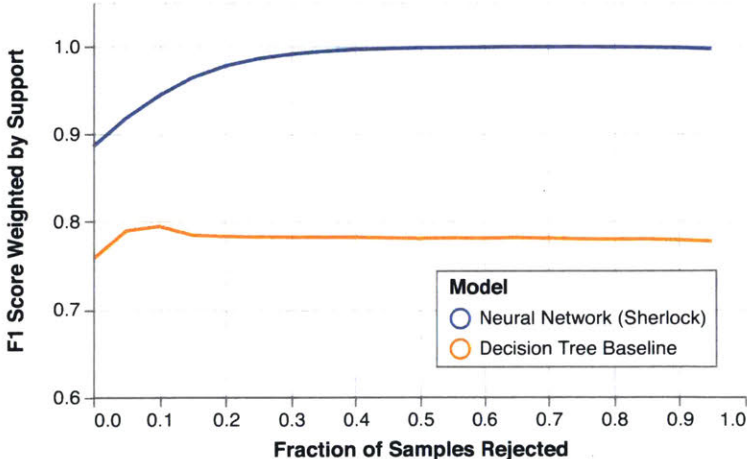


Figure 6-6: Rejection curves showing performance while rejecting all but the top  $x\%$  highest confidence samples.

By introducing a rejection threshold of 10% of the samples, Sherlock reaches an  $F_1$  score of  $\sim 0.95$ . This significant increase in predictive performance suggests a hybrid approach in which low confidence samples are manually annotated. Note that the higher rejection threshold, the lower the error we make in predicting labels, at the cost of needing more expert capacity.

## 6.5 Discussion

We began by considering a set of semantic types described by prior work that identifies the correspondences between DBPedia [7] and WebTables [19]. Then, we constructed a dataset consisting of matches between those types with columns in the VizNet [65] corpus. Inspection of these columns suggests that such an approach yields training samples with few false positives. After extracting four categories of features describing the values of each column, we formulate type detection as a multiclass classification task. A multi-input neural network demonstrates high predictive performance at the classification task compared with the decision tree, matching-based, and crowdsourced benchmarks.

Developers have multiple avenues for incorporating ML-based semantic type detection approaches into systems. To support the use of Sherlock “out-of-the-box,” we distribute Sherlock as a Python library<sup>3</sup> that can be easily installed and incorporated into existing codebases. For developers interested in a different set of semantic types, we open source our training and analysis scripts.<sup>2</sup> The repository also supports developers wishing to retrain Sherlock using data from their specific data ecologies, such as enterprise or research settings with domain-specific data.

## 6.6 Summary and Future Work

Correctly detecting semantic types is critical to many important data science tasks. Machine learning models coupled with large-scale data repositories have demonstrated success across domains, suggesting a promising approach to semantic type detection. Sherlock provides a step forward in this direction.

We identify five promising avenues for future research: (1) enhancing the quantity and quality of the training data, (2) increasing the number of considered types, (3) enriching the set of features extracted from each column, (4) developing shared bench-



marks, and (5) using real-world data repositories to develop systems components that address other parts of the data science workflow.

**Enhancing data quantity and quality.** Machine learning model performance is limited by the number of training examples. Sherlock is no exception. Though the VizNet corpus aggregates datasets from four sources, there is an opportunity to incorporate training examples from additional sources, such as Kaggle,<sup>4</sup> datasets included alongside the R statistical environment,<sup>5</sup> and the ClueWeb web crawl of Excel spreadsheets.<sup>6</sup> We expect increases in the training data diversity to improve the robustness and generalizability of Sherlock.

The quality of model predictions is further determined by the correspondence between training data and unseen testing data, such as the datasets uploaded by analysts to a system. Our method of matching semantic types with columns from real-world data repositories affords both the harvesting of training samples at scale and the ability to use aspects of “dirty” data (e.g., missing values) as features. While we verified the quality of training data through manual inspection, there is an opportunity to label data quality at scale by combining crowdsourcing with active learning. By assessing the quality of each training dataset, this approach would support training semantic type detection models with completely “clean” data at scale.

**Increasing number of semantic types.** To ground our approach in prior work, this chapter considered 78 semantic types described by the T2Dv2 Gold Standard. While 78 semantic types is a substantial increase over what is supported in existing systems, it is a small subset of entities from existing knowledge bases: the DBpedia ontology [7] covers 685 classes, WordNet [182] contains 175K synonym sets, and

---

<sup>4</sup><https://www.kaggle.com/datasets>

<sup>5</sup><https://github.com/vincentarelbundock/Rdatasets>

<sup>6</sup><http://lemurproject.org/clueweb09.php>

Knowledge Graph<sup>7</sup> contains millions of entities. The entities within these knowledge bases, and the hierarchical relationships between entities, provide an abundance of semantic types.

In lieu of a relevant ontology, researchers can count the frequency of column headers in the available data to determine which semantic types to consider. This data-driven approach would ensure the maximum number of training samples for each semantic type. Additionally, these surfaced semantic types are potentially more specific to usecase and data ecology (e.g., data scientists integrating enterprise databases within a company).

**Enriching feature extraction.** We incorporate four categories of features that describe different aspects of individual column values. A promising approach is to include features that describe relationships between columns (e.g., correlation, number of overlapping values, and name similarity), aspects of the entire dataset (e.g., number of columns), and source context (e.g., webpage title for scraped tables). Additionally, although we used features to aid interpretation of results, neural networks using raw data as input are a promising research direction. For example, a character-level recurrent neural network could classify concatenated column values.

**Developing shared benchmarks.** Despite rich prior research in semantic type detection, we could not find a benchmark with publicly available code that accommodates a larger set of semantic types. Therefore, we incorporated benchmarks that approximated state-of-the-art data systems, to the best of our knowledge. However, domains such as image classification and language translation have benefited from shared benchmarks and test sets. Towards this end, we hope that open-sourcing the data and code used in this chapter can benefit future research.

---

<sup>7</sup><https://developers.google.com/knowledge-graph>

**Learning data science systems components.** Semantic type detection is one of many tasks critical to the data science workflow. Prior work has introduced automation into data preparation [73], database query optimization [108], statistical modeling [101], and report generation [40] tasks. Learned systems components addressing these tasks may benefit from approaches such as Sherlock that are trained and benchmarked using repositories of real-world data.

## Chapter 7

### Conclusion

To contend with the increasing volume and availability of data, communities have adopted data visualization as a means to explore and communicate information. Existing data-agnostic visualization systems, which better are suited for tasks such as creating custom visualizations, frequently require programming experience and manual specification. As a result, data visualization remains inaccessible to those without technical experience. This thesis is motivated by the goal of bringing data visualization capabilities to a non-technical audience. One promising approach is visualization recommendation, which automatically generates visualizations for users to search and select. This thesis contributes new systems, methods, and data repositories toward advancing visualization recommendation.

#### 7.1 Review of Contributions

We began by introducing bespoke visualization systems such as Pantheon [205] and the Global Language Network [148]. Reflecting on the challenges of building such systems, we motivated the creation of DIVE, a mixed-initiative data visualization and analysis tool that affords both manual and recommendation-driven data exploration. CHAPTER 3 describes the rule-based recommender system, design considerations, experimental evaluation, and implementation of DIVE. Since its public release in June 2018, DIVE has received thousands of users and has been deployed in both public and commercial settings.

1. **Design considerations** (§3.2): describing the design considerations for mixed-initiative data exploration systems
2. **Open-source system** (§§3.3, 3.4): designing and implementing a publicly avail-

able and open source system that has received over 15K users and was deployed within commercial settings

3. **Evaluation** (§3.5): assessing DIVE for task completion accuracy and time among a group of 67 professional data scientists, compared to an Excel baseline

The limitations of rule-based recommender systems led to VizML, a machine learning-based approach to visualization recommendation using data from a popular online platform. CHAPTER 4 formulates the visualization recommendation problem, describes our collection of a large corpus of datasetvisualization pairs from Plotly, extraction of features from datasets and design choices of visualizations from corresponding visualizations, model training and evaluation, model prediction performance, and crowdsourced benchmarks.

1. **Problem formulation** (§4.1): learning design choices from a corpus of data-visualization pairs
2. **Data processing pipeline** (§§4.2, 4.3): collecting and cleaning corpus, then extracting features and design choices
3. **Predicting design choices** (§4.4): evaluating neural network performance at predicting design choices
4. **Feature importances** (§4.4): reporting and interpreting the contribution of each feature to the prediction tasks
5. **Crowdsourced benchmark** (§4.5): evaluating human and ML models at predicting the crowdsourced visualization type

The lack of high-quality training data in sufficient quantities motivated VizNet, a large-scale visualization learning and benchmarking repository. CHAPTER 5 presents our collection of four corpora from web-based visualization platforms, tables from the web, and open data portals, characterizes the datasets within VizNet, replicates a

prior study using datasets sampled from VizNet, and presents the training of a model to predict the effectiveness of unseen (*dataset, visualization, task*) triplets.

1. **Data** (§5.1): collecting and characterizing four corpora from web-based visualization platforms, tables from the web, and open data portals
2. **Replication** (§§5.2, 5.3): replicating a prior study using the datasets sampled from VizNet
3. **Learning Effectiveness** (§5.3): training a model to predict the effectiveness of unseen (*dataset, visualization, task*) triplets (§5.3)

Enabled by the availability of real-world data, Sherlock is a deep learning approach to semantic type detection. CHAPTER 6 describes the 78 semantic types obtained from the literature, matching column headers from the VizNet corpus to these semantic types, extracting four categories of features that describe the values of each column, training a multi-input neural network, and benchmarking against two matching-based and one crowdsourced benchmark.

1. **Data** (§6.2): Demonstrating a scalable process for matching 686,675 columns from VizNet corpus for 78 semantic types, and then describing each with 1,588 features
2. **Model** (§6.3): Formulating type detection as a multiclass classification problem and then contributing a novel multi-input neural network architecture
3. **Results** (§6.4): Benchmarking predictive performance against a decision tree baseline, two matching-based models, and crowdsourced consensus

## 7.2 Future Work

The contributions – and limitations – of this thesis can hopefully serve as useful starting points for future researchers. Three lines of inquiry are particularly promising because they sit at the intersection of impactful and addressable. The first direction

is to establish design considerations for *mixed-initiative interactions*, which is relevant especially to DIVE. The second is measuring and ensuring the *statistical validity of visual analysis*. This is especially important if ML-based recommender systems gain adoption. The third is assessing the similarities and differences between visualization *expert versus non-expert* judgment, which is relevant to all parts of this thesis, especially VizML. Each section begins by discussing a relevant piece of recent work and then outlines questions and potential approaches for future research.

### 7.2.1 Mixed-Initiative Interactions

Diverse tasks such as spam detection, content recommendation, and cancer screening have benefited from the wholesale replacement of rule-based systems and human judgment with learned systems. These cases of successful automation have heralded efforts to automate the remaining tasks still performed by humans. However, many tasks, especially those requiring domain knowledge, require a more careful balance between human interaction and machine automation.

In the recent paper “Agency plus automation: Designing artificial intelligence into interactive systems,” Jeffrey Heer presents questions for future work along the agency-automation balance. Noting that the present discourse is dominated by artificial intelligence (AI), Heer began by reintroducing the idea of intelligence augmentation (IA) into the design of interactive systems [56]. In particular, IA motivates mixed-initiative systems that “integrate proactive computational support” to augment, instead of replacing, intellectual work. This integration is reified by three systems in the fields of data cleaning, exploratory data analysis, and language translation, each of which possess shared representations of possible actions. To close, Heer raised questions regarding the development and evaluation of mixed-initiative systems that mirror those suggested by this thesis. We discuss three of these questions below.

**Learning from interaction.** The design of current visualization recommendation systems such as DIVE is primarily concerned with how to present recommendations to users. But these recommendations do not improve with continual user interaction. Indeed, they do not “close the loop” by learning from user interaction. Going forward, developing optimal methods that account for user feedback and designing interfaces that account for these methods will be crucial to the effectiveness of visualization recommendation systems.

Distinguishing between Top-1 and Top-N recommendations gives us a means to understand different forms of user interaction with recommendations. Authoring interfaces such as Tableau [173] and Charticulator [146] provide a single-view of a visualization, which can be thought of as Top-1 recommendation. Systems with Top-1 recommendations automatically suggest a strong initial visualization and then continues to make adaptive suggestions as users specify encodings. This interaction is analogous to Google Auto Reply combined with multi-suggestion autocomplete.

Several tools ( [66,119,128,200]) provide a gallery of potential results, analogous to a list of content recommendations on streaming platforms such as Youtube or Spotify. In this general case of Top-N recommendation, a first approach to learning from interaction would let users give binary feedback explicitly by “starring” recommendations or by using up/down voting. Extending this binary boolean feedback into continuous numeric feedback, which is used by the data preparation tool Data Tamer [174], lets annotators supply a confidence or score to a recommendation.

However, harvesting explicit feedback requires additional steps from the user. In contrast, mining implicit feedback from user interactions could provide timely and abundant training data that corresponds more closely to the user’s desires. For example, engaging with a recommendation by “drilling down” into details, an extended mouse hover, or an eye gaze can be interpreted as a form of feedback.



Optimizing Top-N rankings from feedback lends itself to the rule-based content ranking algorithms used on social news aggregators, which originated with Digg and recently used by sites like Hacker News and Reddit. This problem is also addressed by *Learning-to-Rank* methods [98], which, despite potential bias issues [70], remain state-of-the-art at learning from high-dimensional interaction data.

Learned models require overcoming the cold start problem of providing meaningful results with insufficient information about the user, domain, or data. Current systems bootstrap recommendations using defaults. For example, Voyager and DIVE [66, 200] use summary visualizations. Going forward, we suspect that models such as VizML (CHAPTER 4), used either as a pre-trained model or as a source for transfer learning, can help address the cold start problem.

There remains the issue of gathering enough data to properly train a recommendation model. User interaction, especially when viewed as a form of manual labeling, is sparse and expensive. Active learning, as suggested in the VizNet paper (CHAPTER 5), is a promising path to maximizing the information provided by each user annotation. But by optimizing training signal for the model, active learning increases time cost for the user. One approach is addressing both of the prior issues to separate the user workflow into two stages, the first in which the model queries the user and the second in which the user queries the model. Trade-offs between short-term annotation costs and long-term benefits is an open area of research.

**Interpretable models for recommendations.** The development and deployment of mixed-initiative systems raises questions about how to evaluate and interpret such systems. Regarding visualization recommender systems, the top-level evaluation metric is the value of the recommended visualizations. Researchers have proposed numerous frameworks to evaluate visualizations, ranging from effectiveness and expressiveness criteria, to task-based approaches, to insight-based, to long-term studies.

Researchers have proposed various methods for operationalizing these evaluation criteria through online and offline experiments with human subjects.

As automated visualization techniques gain adoption, users may demand evaluation according to desiderata such as fairness, robustness, and usability, which go beyond expected task performance. Identifying and formalizing the desiderata of data-driven visualization systems is an open area of research. Until then, as Doshi-Velez and Kim noted in “Towards A Rigorous Science of Interpretable Machine Learning,” interpretability serves as a popular fallback for other desiderata [39].

Defined in the context of ML systems as the “ability to explain or to present in understandable terms to a human,” interpretability is often needed in scenarios in which the human’s goal is to gain knowledge or where the system faces multi-objective trade-offs. Data visualization employed in the service of exploratory data analysis fits the former criteria. Data visualization used for communication falls under the second criteria.

The authors proposed a taxonomy of interpretability evaluation: progressing from functionally-grounded evaluations with no humans and proxy tasks, to human-grounded metrics with real humans and simplified tasks, and, finally, to application-grounded evaluation with real humans and real tasks. Note that this progression from concrete to abstract parallels the progression of visualization evaluation frameworks. Going forward, interpretable ML research may provide useful concepts and language for reasoning about automated visualization, and visualization may serve as a well-defined testbed for interpretable ML research.

**Deskilling.** A crucial evaluation criteria of AI-infused visualization systems is whether they promote learning and skill acquisition, as opposed to deskilling users. This question is increasingly pertinent across domains as systems, especially medical technolo-

gies, begin incorporating AI and ML techniques. Data visualization is no exception.

Do automated visualization systems reduce the data literacy of users? To begin investigating this question, it is necessary to first identify and distinguish between the groups of users that are adopting such visualization systems. Data visualization reaches users from a diverse set of backgrounds with a wide range of skills. Visualization recommenders have been motivated, in particular, with the dual goal of lowering the barriers-to-entry for non-technical users and reducing tedium for expert users.

By definition, non-technical users by do not possess the skills needed to use visualization systems that require programming. Only a small number of them have the time or resources to acquire programming skills in the future. Therefore, automated visualization systems strictly increases the visualization capabilities of users. The clearest analogy is the widespread adoption of machine language translation. As of May 2017, Google Translate is being used by over 500 million people daily.<sup>1</sup> Though far from perfect,<sup>2</sup> the present state of machine translation is sufficient for many day-to-day and business use cases where the users would probably not have learned a new language. The risk of machine translation is not one of deskilling, but rather one of drastically incorrect results. The analogous question for automated visualization is one of safety: how can we design systems that promote valid statistical inference?

The effect of automated visualization on the skills of technical users is less clear. Programming interfaces that require input in the form of textual languages to specify visualizations suffer from a wide “gulf of execution” [68]. The users of these programming tools may benefit from a shared representation provided by a mixed-initiative tool, as demonstrated by the success of interactive visual design tools such as Lyra [152], Data Illustrator [100], and Charticator [146]. Additionally, automating

---

<sup>1</sup><https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html>

<sup>2</sup><https://www.theatlantic.com/technology/archive/2018/01/the-shalowness-of-google-translate/551570>

tedious manual specification tasks may upskill users by freeing up the time needed for higher-level reasoning tasks such as assessing perceptual effectiveness and statistical validity. This optimistic view is balanced by a pessimistic counterfactual. Prior research on humanrobot interaction [201] shows that humans grow to trust robots more completely than humans, suggesting that users of automated visualization systems may disengage and be uncritical of recommendations.

## 7.2.2 Statistical Validity of Visual Analysis

Analysts using visualization to explore and confirm hypotheses are at risk of arriving at spurious insights via the multiple comparisons problem (MCP) [207]. But if visual analytics tools are fishing rods for spurious insights, then visualizations recommender systems are deep ocean bottom trawlers that implicitly conduct multiple tests in parallel, rather than in series. The MCP is exacerbated by opaque ML-based recommender systems, in which the number of implicit comparisons is difficult or impossible to track.

VizRec, a “framework for secure data exploration” with visualization recommendation systems, recently was introduced to quantify the statistical significance of recommendations [171]. The authors demonstrate two methodologies for automatically adjusting the significance of recommendations based on the recommendation search space. The first classical statistical method derives bounds on the deviation from expectation of a single visualization using Chernoff bounds, with a given level of control for false positive recommendations and for a given size of the data. These bounds are then combined for multiple visualizations using the union bound.

The second method characterizes the Vapnik-Chervonenkis (VC) dimension of recommenders and then derives a maximum VC dimension guaranteed to meet a desired family-wise error rate (FWER) control level with a given size of data. The authors

also discuss the challenge of expanding beyond a single set of recommendations (also called “one-shot predictions”) to the case of iterative data exploration of multiple sets of recommendations in series. In going so, they demonstrate that the VC dimension approach is “*agnostic* to the adaptive nature of the testing as it accounts *preemptively* for *all* possible evaluations of pairs of visualizations.”

The latter method proposed by VizRec provides a powerful bound VC dimension of recommender systems. By applying VizRec to SeeDB [107], the authors demonstrate how some “top” visualizations are not marked as statistically significant. The method can be applied to other data queries recommenders, such as Data Polygamy [27], but also visual encoding recommenders, such as VizML, as long as there is a defined scoring function and enumerable range space.

Issues arise when the scoring function or the range space is infinite, for example when there is not a pre-defined discretization of continuous features and when the scoring function is ill-defined. Safe use of visualization recommenders in either of these cases depends on the use of a holdout test set. Though holdout sets can significantly reduce the power of statistical tests and they can only be used once for a single test, there is active research into “reusable holdouts” that permit multiple validation steps while controlling for false discovery rates [42].

### **7.2.3 Experts versus Non-experts**

Each of the four contributions in this thesis involve human evaluation: DIVE was evaluated by a large group of professional consultants who were fluent in working with data; VizML, VizNet, and Sherlock recruited crowdsourced workers from Amazon Mechanical Turk. The benefits of utilizing crowdsourcing platforms are well-known. Recruiting evaluators is efficient and cheap relative to recruiting expert evaluators, letting researchers collect a significantly larger quantity of experimental data [58],

which is important for both establishing statistically significant experimental results and training generalizable machine learning models.

There are also many limitations when it comes to relying on crowdsourced evaluation. A major limitation is potentially introducing low-quality data into the experiment. However, this limitation can be addressed in part with proper targeting, pre-screens, attention checks, and post-filtering. Certain experiments, such as those involving visualization authoring with programming interfaces, cannot be conducted with non-experts. The major unknown limitations include the experimental effect of expertise. Visualization experts are assumed to have preferences and actions informed by knowledge of perceptual studies, best practices, and tacit experience. Visualization non-experts, in contrast, do not have such backgrounds.

The recent paper titled “A Heuristic Approach to Value-Driven Evaluation of Visualizations” by Wall et al. creates a heuristic-based evaluation methodology to assess the “value” of interactive visualizations [190]. This notion of value, as proposed by John Stasko in 2014 [170] and referred to as ICE-T, consists of four components corresponding to a visualization’s ability to minimize the time needed to answer questions about the data (T), spur and discover insights (I), convey an overall essence (E), and generate confidence about the data (C).

There are many approaches to evaluating the utility of a visualization, such as task performance benchmarks used in DIVE (CHAPTER 3) and VizNet (CHAPTER 5), insight-based methodologies, and longitudinal evaluations. We focus on the ICE-T framework because of their use of experts (15 participants holding Ph.D.s who conduct research in visualization) to evaluate three visualizations according to well-defined criteria (21 heuristics evaluated using a 7 point likert scale). Assessing inter-rater reliability, Wall et al. found substantial and statistically significant agreement among the raters (correlation coefficient  $r = 0.66$  across all visualizations, with a significance level of  $p < 0.05$  for each.) These results show that experts are consistent amongst

themselves for this specific task with these specific visualizations.

To assess the within-group agreement among non-experts and the between-group agreement, we can conduct the same heuristic-based evaluation among crowdsourced workers. We use the same three visualizations about U.S. colleges with pseudorandom ordering of visualizations, such that all six possible visualization orderings were sampled evenly. We recruited 146 workers from Amazon Mechanical Turk, each of which have a  $> 95\%$  HIT approval rating, are native English speakers, and are using a desktop computer. Then, we filtered out 18 participants whose score variance across all heuristics was less than a threshold of 0.25.

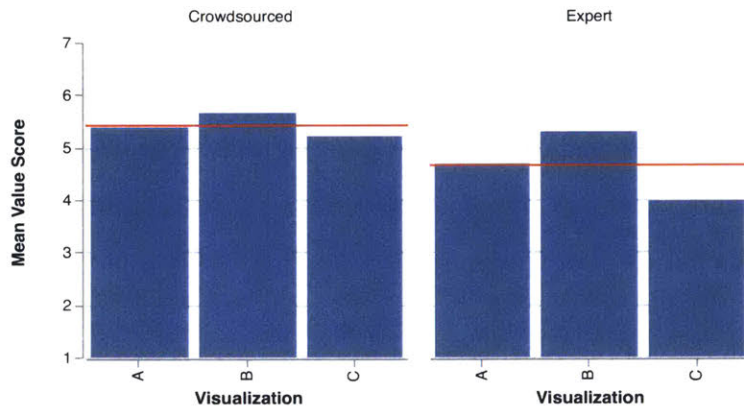


Figure 7-1: Mean visualization value across three visualizations for crowdsourced workers and experts.

We calculate the value of the three visualizations per participant by taking the simple average of the four guideline scores, which are simple averages of the component heuristic scores. The final value of a visualization is the average across all of the participants. As shown in Figure 7-1, the mean of the crowdsourced scores (5.41) is higher than that of the expert scores (4.65), but the relative rank of the visualizations ( $B > A > C$ ) is consistent between the two groups.

Further analyzing these visualizations, we show the correlation between crowdsourced and expert heuristic scores in Figure 7-2. Heuristic scores are not correlated for Visualization A ( $r = 0.17$ ,  $p = 0.46$ ) but are significantly correlated for Visualization B ( $r = 0.55$ ,  $p = 0.009$ ) and Visualization C ( $r = 0.64$ ,  $p = 0.001$ ). In particular,

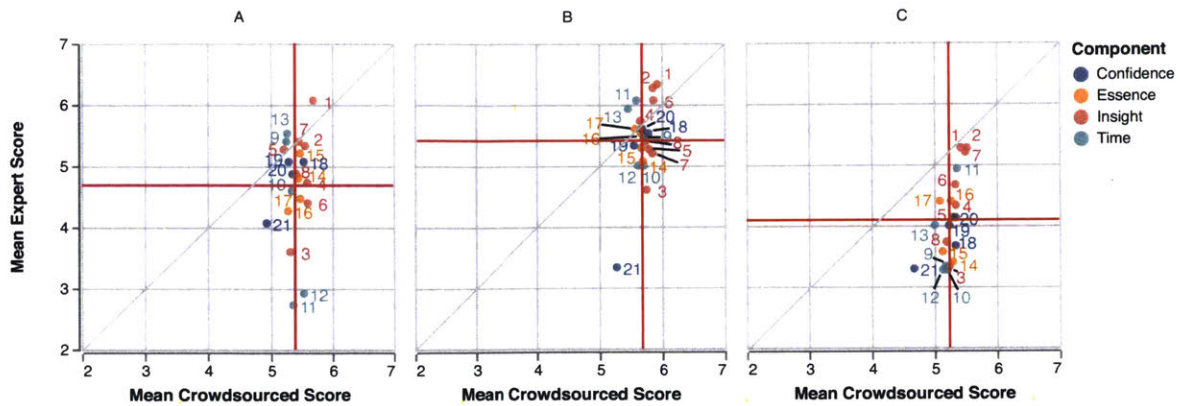


Figure 7-2: Relationship between expert and crowdsourced heuristic scores

for Visualization A, the two groups disagree strongly on heuristics 11 (“The interface supports reorganizing the visualization by the data’s attribute values”) and 12 (“The visualization supports smooth transitions between different levels of detail in viewing the data”). Taken together, these results suggest that the judgment of crowdsourced workers may align with expert judgment in aggregate, but that there remain large departures between the two for heuristic-level judgments.

### 7.3 Closing Remarks

*In a properly automated and educated world, then, machines may prove to be the true humanizing influence. It may be that machines will do the work that makes life possible and that human beings will do all the other things that make life pleasant and worthwhile*

*Robot Visions*

ISAAC ASIMOV, 1990

The volume of data and demand for data-derived insights is exponentially across domains. Data visualization is one means to explore and communicate such insights. Automated visualization is a promising means for bringing visualization capabilities to domain experts who do not necessarily have technical backgrounds. In particular,



visualization recommendation endeavors to minimize the labor needed to create visualizations while respecting user agency. By contributing systems, methods, and data for visualization recommendation, this thesis hopes to make visualization a capability available to all those with data, not only those with technical backgrounds.

## Bibliography

- [1] Martín Abadi et al. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.
- [2] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st edition, 2016.
- [3] Ashok K. Agrawala. Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, 16(4):373–379, July 1970.
- [4] Christopher Ahlberg. Spotfire: An Information Exploration Environment. *SIGMOD Rec.*, 25(4):25–29, December 1996.
- [5] Christopher Ahlberg and Ben Shneiderman. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '94*, pages 313–317, New York, NY, USA, 1994. ACM.
- [6] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *Proceedings - IEEE Symposium on Information Visualization, INFO VIS*, 2005.
- [7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. pages 722–735, 2007.
- [8] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1217–1230, 2016.
- [9] Leilani Battle, Remco Chang, Jeffrey Heer, and Michael Stonebraker. Position statement: The case for a visualization performance benchmark. In *2017 IEEE Workshop on Data Systems for Interactive Analysis (DSIA)*, pages 1–5, Oct 2017.
- [10] Leilani Battle, Peitong Duan, Zachery Miranda, Dana Mukusheva, Remco Chang, and Michael Stonebraker. Beagle: Automated Extraction and Interpretation of Visualizations from the Web. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18*, pages 594:1–594:8, New York, NY, USA, 2018. ACM.
- [11] Jacques Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

- [12] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.
- [13] Michelle A Borkin, Azalea A Vo, Zoya Bylinskii, Phillip Isola, Shashank Sunkavalli, Alfonso Oliva, and Hanspeter Pfister. What makes a visualization memorable? *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2306–2315, 2013.
- [14] Katy Börner, Adam Maltese, Russell Nelson Balliet, and Joe Heimlich. Investigating aspects of data visualization literacy using 20 information visualizations and 273 science museum visitors. *Information Visualization*, 15(3):198–213, 2016.
- [15] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, December 2011.
- [16] Leo Breiman, Jerome Friedman, R. A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.
- [17] Erik Brynjolfsson and Kristina McElheran. The Rapid Adoption of Data-Driven Decision-Making. *American Economic Review*, 106(5):133–39, May 2016.
- [18] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.
- [19] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. WebTables: Exploring the Power of Tables on the Web. *Proc. VLDB Endow.*, 1(1):538–549, August 2008.
- [20] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
- [21] Stephen M. Casner. Task-analytic Approach to the Automated Design of Graphic Presentations. *ACM Trans. Graph.*, 10(2):111–151, April 1991.
- [22] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. Aurum: A data discovery system. pages 1001–1012, 04 2018.

- [23] Raul Castro Fernandez, Essam Mansour, Abdulhakim Qahtan, Ahmed Elmagarmid, Ihab Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. 04 2018.
- [24] Nick Cawthon and Andrew Vande Moere. The Effect of Aesthetic on the Usability of Data Visualization. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 637–648, July 2007.
- [25] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.
- [26] Zhe Chen, Sasha Dadiomov, Richard Wesley, Gang Xiao, Daniel Cory, Michael Cafarella, and Jock Mackinlay. Spreadsheet property detection with rule-assisted active learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17*, pages 999–1008, New York, NY, USA, 2017. ACM.
- [27] Fernando Chirigati, Harish Doraiswamy, Theodoros Damoulas, and Juliana Freire. Data polygamy: The many-many relationships among urban spatio-temporal data sets. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD '16*, pages 1011–1025, New York, NY, USA, 2016. ACM.
- [28] William S. Cleveland. *Visualizing Data*. Hobart Press, 1993.
- [29] William S. Cleveland, Persi Diaconis, and Robert McGill. Variables on scatterplots look more highly correlated when the scales are increased. Technical report, DTIC Document, 1982.
- [30] 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.
- [31] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.
- [32] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756, 2010.
- [33] Andrew M. Dai, Christopher Olah, and Quoc V. Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015.

- [34] Çagatay Demiralp, Peter J. Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. Foresight: Rapid Data Exploration Through Guideposts. *CoRR*, abs/1709.10513, 2017.
- [35] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.
- [36] Victor Dibia and Çagatay Demiralp. Data2Vis: Automatic Generation of Data Visualizations Using Sequence to Sequence Recurrent Neural Networks. *CoRR*, abs/1804.03126, 2018.
- [37] DOMO. Data never sleeps. Technical report, DOMO, 2018.
- [38] John Qi Dong and Chia-Han Yang. Business value of big data analytics: A systems-theoretic approach and empirical test. *Information and Management*, 2018.
- [39] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. 2017.
- [40] Longxu Dou, Guanghui Qin, Jinpeng Wang, Jin-Ge Yao, and Chin-Yew Lin. Data2text studio: Automated text generation from structured data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 13–18, Brussels, Belgium, November 2018. Association for Computational Linguistics.
- [41] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. Shepherd-ing the crowd yields better work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1013–1022. ACM, 2012.
- [42] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2350–2358. Curran Associates, Inc., 2015.
- [43] Humaira Ehsan, Mohamed A. Sharaf, and Panos K. Chrysanthis. MuVE: Efficient Multi-Objective View Recommendation for Visual Data Exploration. *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 731–742, 2016.
- [44] Philipp Eichmann, Carsten Binnig, Tim Kraska, and Emanuel Zraggen. IDEBench: A Benchmark for Interactive Data Exploration. *ArXiv e-prints*, April 2018.

- [45] Stephen G. Eick, Joseph L. Steffen, and Eric E. Sumner Jr. Seesoft—a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, Nov 1992.
- [46] Jean-Daniel Fekete. The infovis toolkit. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pages 167–174, Washington, DC, USA, 2004. IEEE Computer Society.
- [47] Stephen Few. Data Visualization Effectiveness Profile. [https://www.perceptualedge.com/articles/visual\\_business\\_intelligence/data\\_visualization\\_effectiveness\\_profile.pdf](https://www.perceptualedge.com/articles/visual_business_intelligence/data_visualization_effectiveness_profile.pdf), 2017.
- [48] Snehal Neil Gaikwad, Durim Morina, Rohit Nistala, Megha Agarwal, Alison Cossette, Radhika Bhanu, Saiph Savage, Vishwajeet Narwal, Karan Rajpal, Jeff Regino, et al. Daemo: A self-governed crowdsourcing marketplace. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 101–102. ACM, 2015.
- [49] Snehal Neil S Gaikwad, Durim Morina, Adam Ginzberg, Catherine Mullings, Shirish Goyal, Dilrukshi Gamage, Christopher Diemert, Mathias Burton, Sharon Zhou, Mark Whiting, et al. Boomerang: Rebounding the consequences of reputation feedback on crowdsourcing platforms. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 625–637. ACM, 2016.
- [50] Gapminder Foundation. Gapminder. <http://www.gapminder.org>, 2017.
- [51] Aman Goel, Craig A Knoblock, and Kristina Lerman. Exploiting structure within data for accurate labeling using conditional random fields. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, 2012.
- [52] Google. Explore in Google Sheets. <https://www.youtube.com/watch?v=9TiXR5wwqPs>, 2015.
- [53] Google. Google Data Studio. <https://datastudio.google.com>, 2016.
- [54] Christopher Groskopf and contributors. csvkit. <https://csvkit.readthedocs.org>, 2016.
- [55] Frederick Hayes-Roth. Rule-based Systems. *Commun. ACM*, 28(9):921–932, September 1985.
- [56] Jeffrey Heer. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences*, 116(6):1844–1850, 2019.

- [57] Jeffrey Heer and Maneesh Agrawala. Multi-scale banking to 45 degrees. *IEEE Trans. Visualization & Comp. Graphics*, 12:701–708, 2006.
- [58] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: Using mechanical turk to assess visualization design. In *ACM Human Factors in Computing Systems (CHI)*, 2010.
- [59] Jeffrey Heer, Stuart K. Card, and James A. Landay. Prefuse: A toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 421–430, New York, NY, USA, 2005. ACM.
- [60] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1303–1312, New York, NY, USA, 2009. ACM.
- [61] Jeffrey Heer and Ben Shneiderman. Interactive dynamics for visual analysis. *Commun. ACM*, 55(4):45–54, April 2012.
- [62] Jeffrey Heer, Fernanda Viégas, and Martin Wattenberg. Voyagers and voyeurs: Supporting asynchronous collaborative information visualization. In *ACM Human Factors in Computing Systems (CHI)*, pages 1029–1038, 2007.
- [63] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 159–166, New York, NY, USA, 1999. ACM.
- [64] Kevin Hu, Michiel 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.
- [65] Kevin Hu, Neil Gaikwad, Michiel Bakker, Madelon Hulsebos, Emanuel Zraggen, César Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çağatay Demiralp. Viznet: Towards a large-scale visualization learning and benchmarking repository. In *Proceedings of the 2019 Conference on Human Factors in Computing Systems (CHI)*. ACM, 2019.
- [66] Kevin Hu, Diana Orghian, and César Hidalgo. DIVE: A Mixed-Initiative System Supporting Integrated Data Exploration Workflows. In *ACM SIGMOD Workshop on Human-in-the-Loop Data Analytics (HILDA)*. ACM, 2018.

- [67] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zraggen, Arvind Satyanarayan, Tim Kraska, Çağatay Demiralp, , and César Hidalgo. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *Proceedings of the 2019 Conference on Knowledge Discovery and Data Mining (CHI)*. ACM, 2019.
- [68] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct manipulation interfaces. *Hum.-Comput. Interact.*, 1(4):311–338, December 1985.
- [69] Shinobu Ishihara. *Tests for colour-blindness*. Kanehara Shuppan Company, 1960.
- [70] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17*, pages 781–789, New York, NY, USA, 2017. ACM.
- [71] Eric McKenzie Jonathan Meddes. Improving visualization by capturing domain knowledge. volume 3960, pages 3960 – 3960 – 10, 2000.
- [72] Ben Jones. Data Dialogues: To Optimize or to Satisfice When Visualizing Data? <https://www.tableau.com/about/blog/2016/1/data-dialogues-optimize-or-satisfice-data-visualization-48685>, 2016.
- [73] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *ACM Human Factors in Computing Systems (CHI)*, 2011.
- [74] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, December 2012.
- [75] Helen Kennedy, Rosemary Lucy Hill, William Allen, and Andy Kirk. In *Engaging with (big) data visualizations: Factors that affect engagement and resulting new definitions of effectiveness*, volume 21, USA, 2016. First Monday.
- [76] Tom Kenter and Maarten De Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1411–1420. ACM, 2015.
- [77] Annice E. Kim, Heather M. Hansen, Joe Murphy, Ashley K. Richards, Jennifer Duke, and Jane A. Allen. Methodological considerations in analyzing twitter data. *JNCI Monographs*, 2013(47):140–146, 2013.
- [78] Younghoon Kim and Jeffrey Heer. Assessing Effects of Task and Data Distribution on the Effectiveness of Visual Encodings. *Computer Graphics Forum (Proc. EuroVis)*, 2018.



- [79] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.
- [80] Aniket Kittur, Jeffrey V. Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 1301–1318. ACM, 2013.
- [81] Cole Nussbaumer Knaflic. Is there a single right answer? <http://www.storytellingwithdata.com/blog/2016/1/12/is-there-a-single-right-answer>, 2016.
- [82] Nicholas Kong, Jeffrey Heer, and Maneesh Agrawala. Perceptual guidelines for creating rectangular treemaps. *IEEE Trans. Visualization & Comp. Graphics*, 16(6):990–998, 2010.
- [83] Robert Kosara. Understanding Pie Charts. <https://eagereyes.org/techniques/pie-charts>, 2010.
- [84] Robert Kosara and Drew Skau. Judgment error in pie chart variations. In *Proceedings of the Eurographics/IEEE VGTC Symposium on Visualization*, pages 91–95. Wiley Online Library, 2016.
- [85] Clemens Kruse, Rishi Goswamy, Yesha Raval, and Sarah Marawi. Challenges and opportunities of big data in health care: A systematic review. *JMIR Medical Informatics*, 4:e38, 11 2016.
- [86] Bum chul Kwon, Brian Fisher, and Ji Soo Yi. Visual analytic roadblocks for novice investigators. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 3–11, Oct 2011.
- [87] Open Knowledge Labs. messytables. <https://github.com/okfn/messytables>, 2013.
- [88] Heidi Lam, Tamara Munzner, and Robert Kincaid. Overview use in multiple visual information resolution interfaces. *IEEE Trans. Visualization & Comp. Graphics*, 13(6):1278–1285, 2007.
- [89] John Le, Andy Edmonds, Vaughn Hester, and Lukas Biewald. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR 2010 workshop on crowdsourcing for search evaluation*, volume 2126, 2010.
- [90] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

- [91] Jeehyung Lee, Wipapat Kladwang, Minjae Lee, Daniel Cantu, Martin Azizyan, Hanjoo Kim, Alex Limpaecher, Snehal Gaikwad, Sungroh Yoon, Adrien Treuille, et al. Rna design rules from a massive open laboratory. *Proceedings of the National Academy of Sciences*, 111(6):2122–2127, 2014.
- [92] Stephan Lewandowsky and Ian Spence. Discriminating strata in scatterplots. *Journal of American Statistical Association*, 84(407):682–688, 1989.
- [93] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49:764766, 07 2013.
- [94] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3(1-2):1338–1347, 2010.
- [95] Chin-Yew Lin. ROUGE: a package for automatic evaluation of summaries. In *ACL 2004*, pages 25–26, 2004.
- [96] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [97] Chris J. Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan Andreescu, et al. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.
- [98] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.
- [99] Yang Liu and Jeffrey Heer. Somewhere Over the Rainbow: An Empirical Assessment of Quantitative Colormaps. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 598:1–598:12, New York, NY, USA, 2018. ACM.
- [100] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 123:1–123:13, New York, NY, USA, 2018. ACM.

- [101] James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. *CoRR*, abs/1402.4304, April 2014.
- [102] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding Variable Importances in Forests of Randomized Trees. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’13, pages 431–439, USA, 2013. Curran Associates Inc.
- [103] Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. DeepEye: Towards Automatic Data Visualization. *The 34th IEEE International Conference on Data Engineering (ICDE)*, 2018.
- [104] Alan M. MacEachren. *How Maps Work: Representation, Visualization, and Design*. Guilford Press, 1995.
- [105] Jock Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.
- [106] 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, November 2007.
- [107] Aditya Parameswaran Manasi Vartak, Samuel Madden and Neoklis Polyzotis. SeeDB: Automatically Generating Query Visualizations. *Proceedings of the VLDB Endowment*, 7(13):1581–1584, 2014.
- [108] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: A Learned Query Optimizer. *arXiv e-prints*, page arXiv:1904.03711, Apr 2019.
- [109] Winter Mason and Siddharth Suri. Conducting behavioral research on amazon’s mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.
- [110] Frank J. Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.
- [111] Michele Mauri, Tommaso Elli, Giorgio Caviglia, Giorgio Ubaldi, and Matteo Azzi. Rawgraphs: A visualisation platform to create open outputs. In *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter, CHIItaly ’17*, pages 28:1–28:5, New York, NY, USA, 2017. ACM.
- [112] Andrew McAfee and Erik Brynjolfsson. Big data: The management revolution. *Harvard business review*, 90:60–6, 68, 128, 10 2012.

- [113] Matthew J. Menne, Imke Durre, Russell S. Vose, Byron E. Gleason, and Tamara G. Houston. An overview of the global historical climatology network-daily database. *Journal of Atmospheric and Oceanic Technology*, 29(7):897–910, 2012.
- [114] Microsoft. Microsoft Excel.
- [115] Microsoft. Power BI — Interactive Data Visualization BI. <https://powerbi.microsoft.com>, 2019.
- [116] Patrick Millais, Simon L. Jones, and Ryan Kelly. Exploring Data in Virtual Reality: Comparisons with 2D Data Visualizations. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, pages LBW007:1–LBW007:6, New York, NY, USA, 2018. ACM.
- [117] George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
- [118] Johann Mitlohner, Sebastian Neumaier, Jurgen Umbrich, and Axel Polleres. Characteristics of open data csv files. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 72–79, Aug 2016.
- [119] 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 Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2018.
- [120] Kristi Morton, Magdalena Balazinska, Dan Grossman, Robert Kosara, and Jock Mackinlay. Public data and visualizations: How are many eyes and tableau public used for collaborative analytics? *SIGMOD Record*, 43(2):17–22, 6 2014.
- [121] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with Noisy Labels. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 1196–1204, USA, 2013. Curran Associates Inc.
- [122] Sebastian Neumaier, Jürgen Umbrich, and Axel Polleres. Automated Quality Assessment of Metadata across Open Data Portals. *Journal of Data and Information Quality*, 2016.
- [123] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

- [124] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.
- [125] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.
- [126] Roger D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.
- [127] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [128] Daniel B. Perry, Bill Howe, Alicia M. F. Key, and Cecilia Aragon. Vizdeck: Streamlining exploratory visual analytics of scientific data. In *iConference*, 2013.
- [129] Minh Pham, Suresh Alse, Craig A Knoblock, and Pedro Szekely. Semantic labeling: a domain-independent approach. In *International Semantic Web Conference*, pages 446–462. Springer, 2016.
- [130] Plotly. Plotly. <https://plot.ly>, 2018.
- [131] Plotly. Plotly Chart Studio. <https://plot.ly/online-chart-maker/>, 2018.
- [132] Plotly. Plotly Community Feed. <https://plot.ly/feed>, 2018.
- [133] Plotly. Plotly for Python. <https://plot.ly/d3-js-for-python-and-pandas-charts/>, 2018.
- [134] Plotly. Plotly REST API. <https://api.plot.ly/v2>, 2018.
- [135] Plotly. Plotly.js Open-Source Announcement. <https://plot.ly/javascript/open-source-announcement>, 2018.
- [136] Nikhil Waman Puranik. A specialist approach for classification of column data. Master’s thesis, University of Maryland, Baltimore County, August 2012.
- [137] Qlik. Qlik Sense. <http://www.qlik.com/us/products/qlik-sense>, 2017.

- [138] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.
- [139] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics, 2016.
- [140] Vijayshankar Raman and Joseph M. Hellerstein. Potter’s wheel: An interactive data cleaning system. In *Proceedings of the 27th International Conference on Very Large Data Bases, VLDB ’01*, pages 381–390, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
- [141] S. Krishnamurthy Ramnandan, Amol Mittal, Craig A Knoblock, and Pedro Szekely. Assigning semantic labels to data sources. In *European Semantic Web Conference*, pages 403–417. Springer, 2015.
- [142] Ernesto Ramos and David Donoho. ASA Data Exposition Dataset. <http://stat-computing.org/dataexpo/1983.html>, 1983.
- [143] Ramana Rao and Stuart K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *ACM Human Factors in Computing Systems (CHI)*, pages 318–322, 1994.
- [144] Khairi Reda, Pratik Nalawade, and Kate Ansah-Koi. Graphical Perception of Continuous Quantitative Maps: The Effects of Spatial Frequency and Colormap Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI ’18*, pages 272:1–272:12, New York, NY, USA, 2018. ACM.
- [145] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA, May 2010.
- [146] Donghao Ren, Bongshin Lee, and Matthew Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics*, January 2019.
- [147] Dominique Ritze and Christian Bizer. Matching web tables to DBpedia – a feature utility study. *context*, 42(41):19, 2017.
- [148] Shahar Ronen, Bruno Gonçalves, Kevin Z. Hu, Alessandro Vespignani, Steven Pinker, and César A. Hidalgo. Links that speak: The global language network and its association with global fame. *Proceedings of the National Academy of Sciences*, 111(52):E5616–E5622, 2014.

- [149] Steven F. Roth, John Kolojechick, Joe Mattis, and Jade Goldstein. Interactive graphic design using automatic presentation knowledge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 112–117, New York, NY, USA, 1994. ACM.
- [150] Bahador Saket, Alex Endert, and Çagatay Demiralp. Task-Based Effectiveness of Basic Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2018.
- [151] Beatriz Sousa Santos. Evaluating Visualization Techniques and Tools: What Are the Main Issues. In *the 2008 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods For information Visualization (BELIV'08)*, 2008.
- [152] Arvind Satyanarayan and Jeffrey Heer. Lyra: An interactive visualization design environment. *Computer Graphics Forum (Proc. EuroVis)*, 2014.
- [153] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, January 2017.
- [154] Arvind Satyanarayan, Kanit Wongsuphasawat, and Jeffrey Heer. Declarative Interaction Design for Data Visualization. In *ACM User Interface Software & Technology (UIST)*, 2014.
- [155] Christoph Schulz, Arlind Nocaj, Mennatallah El-Assady, Steffen Frey, Marcel Hlawatsch, Michael Hund, Grzegorz Karch, Rudolf Netzel, Christin Schätzle, Miriam Butt, et al. Generative data models for validation and evaluation of visualization techniques. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, pages 112–124. ACM, 2016.
- [156] Marc M. Sebrechts, John V. Cugini, Sharon J. Laskowski, Joanna Vasilakis, and Michael S. Miller. Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 3–10, New York, NY, USA, 1999. ACM.
- [157] Michael Sedlmair, Andrada Tatu, Tamara Munzner, and Melanie Tory. A taxonomy of visual cluster separation factors. In *Computer Graphics Forum*, volume 31, pages 1335–1344, 2012.
- [158] Edward Segel and Jeffrey Heer. Narrative Visualization: Telling Stories with Data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, November 2010.

- [159] Jinwook Seo and Ben Shneiderman. A Rank-by-Feature Framework for Interactive Exploration of Multidimensional Data. *Information Visualization*, 4:96–113, 2005.
- [160] Po shen Lee, Jevin D. West, and Bill Howe. Viziometrics: Analyzing visual information in the scientific literature. *IEEE Transactions on Big Data*, 4(1):117–129, 2018.
- [161] Barbara G. Shortridge. Stimulus processing models from psychology: can we use them in cartography? *The American Cartographer*, 9:155–167, 1982.
- [162] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya G. Parameswaran. zenvisage: Effortless visual data exploration. *CoRR*, abs/1604.03583, 2016.
- [163] Samuel Silva, Beatriz Sousa Santos, and Joaquim Madeira. Using color in visualization: A survey. *Computers & Graphics*, 35(2):320 – 333, 2011. Virtual Reality in Brazil Visual Computing in Biology and Medicine Semantic 3D media and content Cultural Heritage.
- [164] David Simkin and Reid Hastie. An information-processing analysis of graph perception. *Journal of American Statistical Association*, 82(398):454–465, 1987.
- [165] Alexander J. G. Simoes and César A. Hidalgo. The economic complexity observatory: An analytical tool for understanding the dynamics of economic development. In *Proceedings of the 17th AAAI Conference on Scalable Integration of Analytics and Visualization*, AAAIWS’11-17, pages 39–42. AAAI Press, 2011.
- [166] Drew Skau. Best Practices: Maximum Elements For Different Visualization Types. <https://visual.ly/blog/maximum-elements-for-visualization-types/>, 2012.
- [167] Drew Skau and Robert Kosara. Arcs, angles, or areas: Individual data encodings in pie and donut charts. In *Computer Graphics Forum*, volume 35, pages 121–130. Wiley Online Library, 2016.
- [168] Ian Spence and Stephan Lewandowsky. Displaying proportions and percentages. *Applied Cognitive Psychology*, 5:61–77, 1991.
- [169] Jennifer G. Stadler, Kipp Donlon, Jordan D. Siewert, Tessa Franken, and Nathaniel E. Lewis. Improving the efficiency and ease of healthcare analysis through use of data visualization dashboards. *Big Data*, 4(2):129–135, 2016.
- [170] John Stasko. Value-driven evaluation of visualizations. In *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, BELIV ’14, pages 46–53, New York, NY, USA, 2014. ACM.



- [171] Lorenzo De Stefani, Leonhard F. Spiegelberg, Tim Kraska, and Eli Upfal. Vizrec: A framework for secure data exploration via visual representation. *CoRR*, abs/1811.00602, 2018.
- [172] Stanley Smith Stevens. On the Theory of Scales of Measurement. *Science, New Series*, 103(2684):677–680, 1946.
- [173] Chris Stolte and Pat Hanrahan. Polaris: A system for query, analysis and visualization of multi-dimensional relational databases. In *Proceedings of the IEEE Symposium on Information Visualization 2000*, INFOVIS '00, pages 5–, Washington, DC, USA, 2000. IEEE Computer Society.
- [174] Michael Stonebraker, Daniel Bruckner, Ihab F. Ilyas, George Beskales, Mitch Cherniack, Stanley B. Zdonik, Alexander Pagan, and Zhan Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.
- [175] Zareen Syed, Tim Finin, Varish Mulwad, Anupam Joshi, et al. Exploiting a web of semantic data for interpreting tables. In *Proceedings of the Second Web Science Conference*, 2010.
- [176] Justin Talbot, John Gerth, and Pat Hanrahan. Arc length-based aspect ratio selection. *IEEE Trans. Visualization & Comp. Graphics*, 2011.
- [177] Justin Talbot, Sharon Lin, and Pat Hanrahan. An extension of Wilkinson’s algorithm for positioning tick labels on axes. *IEEE Trans. Visualization & Comp. Graphics*, 2010.
- [178] Justin Talbot, Vidya Setlur, and Anushka Anand. Four experiments on the perception of bar charts. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2152–2160, Dec 2014.
- [179] Lothar Tremmel. The visual separability of plotting symbols in scatterplots. *Journal of Computational and Graphical Statistics*, 4(2):101–112, 1995.
- [180] Trifacta. Data Wrangling Tools & Software. <https://www.trifacta.com>, 2019.
- [181] John Wilder Tukey. *Exploratory Data Analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company, 1977.
- [182] Princeton University. About WordNet. <https://wordnet.princeton.edu>, 2010.
- [183] Stef van den Elzen and Jarke J. van Wijk. Small multiples, large singles: A new approach for visual data exploration. In *Proceedings of the 15th Eurographics Conference on Visualization*, EuroVis '13, pages 191–200, Chichester, UK, 2013. The Eurographs Association &#38; John Wiley &#38; Sons, Ltd.

- [184] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. Towards visualization recommendation systems. *SIGMOD Rec.*, 45(4):34–39, May 2017.
- [185] Vega. Datalib. <http://vega.github.io/datalib>, 2017.
- [186] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 4(9):528–538, 2011.
- [187] Fernanda Viégas, Martin Wattenberg, Daniel Smilkov, James Wexler, and Daniel Gundry. Generating charts from data in a data table. US 20180088753 A1., 2018.
- [188] Fernanda B. Viégas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. ManyEyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, November 2007.
- [189] Luis Von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.
- [190] Emily Wall, Meeshu Agnihotri, Laura E. Matzen, Kristin Divis, Michael J. Haass, Alex Endert, and John T. Stasko. A heuristic approach to value-driven evaluation of visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25:491–500, 2018.
- [191] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- [192] Martin Wattenberg. Baby Names, Visualization, and Social Data Analysis. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 1–, Washington, DC, USA, 2005. IEEE Computer Society.
- [193] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer Publishing Company, Incorporated, 2nd edition, 2009.
- [194] Hadley Wickham. Tidy data. *The Journal of Statistical misc*, 59, 2014.
- [195] Leland Wilkinson. *The Grammar of Graphics*. Springer, 2nd edition, 2005.
- [196] Leland Wilkinson, Anushka Anand, and Robert Grossman. Graph-theoretic scagnostics. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 21–, Washington, DC, USA, 2005. IEEE Computer Society.

- [197] Graham Wills and Leland Wilkinson. AutoVis: Automatic Visualization. *Information Visualization*, 9:47–6927, 2010.
- [198] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Towards a general-purpose query language for visualization recommendation. In *ACM SIGMOD Human-in-the-Loop Data Analysis (HILDA)*, 2016.
- [199] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2016.
- [200] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *ACM Human Factors in Computing Systems (CHI)*, 2017.
- [201] Jane Wu, Erin Paeng, Kari Linder, Piercarlo Valdesolo, and James C. Boerkoel. Trust and cooperation in human-robot decision making. In *AAAI Fall Symposia*, 2016.
- [202] Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *Int. J. Comput. Vision*, 2016.
- [203] Mehmet Adil Yaln, Niklas Elmqvist, and Benjamin B. Bederson. Keshif: Rapid and expressive tabular data exploration for novices. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2017.
- [204] Cong Yan and Yeye He. Synthesizing type-detection logic for rich semantic data types using open-source code. In *Proceedings of the 2018 International Conference on Management of Data*, pages 35–50. ACM, 2018.
- [205] Amy Zhao Yu, Kevin Zeng Hu, Deepak Jagdish, and César A. Hidalgo. Pantheon: Visualizing historical cultural production. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 289–290, Oct 2014.
- [206] Benjamin Zapilko, Matthäus Zloch, and Johann Schaible. Utilizing regular expressions for instance-based schema matching. volume 946, 11 2012.
- [207] Emanuel Zraggen, Zheguang Zhao, Robert Zeleznik, and Tim Kraska. Investigating the Effect of the Multiple Comparisons Problem in Visual Analysis . In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 479:1–479:12, New York, NY, USA, 2018. ACM.

- [208] Ying Zhu. Measuring Effective Data Visualization . In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Nikos Paragios, Syeda-Mahmood Taveer, Tao Ju, Zicheng Liu, Sabine Coquillart, Carolina Cruz-Neira, Torsten Müller, and Tom Malzbender, editors, *Advances in Visual Computing*, pages 652–661, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.