Techniques for Modifying and Augmenting Existing Charts for Improved Usability

Nicholas Kong

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-212
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-212.html

December 16, 2013

Techniques for Modifying and Augmenting Existing Charts for Improved Usability

By

Nicholas Chi-Yuen Kong

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Maneesh Agrawala, Chair
Professor Björn Hartmann
Professor Marti Hearst
Professor Joseph Hellerstein

Fall 2013

Techniques for Modifying and Augmenting Existing Charts for Improved Usability

Abstract

Techniques for Modifying and Augmenting Existing Charts for Improved Usability

by

Nicholas Chi-Yuen Kong

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Maneesh Agrawala, Chair

Charts abound in printed media and on the web. They have become a primary means to communicate quantitative information. Furthermore, the rise of visualization authoring packages, such as D3, mean creating visualizations is easier than ever before. However, the problem of creating *effective, usable* visualizations remains challenging.

The effectiveness of a visualization is tied to both its design and the task a viewer wishes to complete. Fortunately, researchers have made great progress in the past decades in developing empirically derived design guidelines for visualizations. Some have explored the effectiveness of design choices through graphical perception experiments. Others have outlined the perceptual and cognitive processes viewers undergo when completing a task with a chart. Together, these strands of research provide a foundation for good visualization design.

Yet, many visualizations do not adhere to these guidelines. Some may have been created before such guidelines were available, while others may simply ignore them. In addition, the usability of a visualization may differ from viewer to viewer, depending on the specific task she wishes to complete. In this thesis, we explore techniques for improving the usability of existing, bitmap visualizations by modifying or augmenting them.

In order to modify charts, we need access to the locations of the marks in the chart (e.g., the bars in a bar chart, or the slices in a pie chart) and the underlying data. We present algorithms that automatically extract this information from raster bar and pie charts that obey some common assumptions. Using a corpus of images drawn from the web, these algorithms successfully extract marks from 79% of bar charts and 62% of pie charts, and from these charts they successfully extract the data from 71% of bar charts and 64% of pie charts. We then present an application that uses the extracted marks and data to present a gallery of redesigns.

Next, we tackle the problem of customizing existing visualizations to best support a viewer's goal. Reading a visualization can involve a number of tasks such as extracting, comparing or aggregating numerical values, but most visualizations only support a subset of these tasks. We introduce graphical overlays—visual elements that are layered onto charts

to facilitate a larger set of chart reading tasks. These overlays directly support the lower-level perceptual and cognitive processes that viewers must perform to read a chart. We identify five main types of overlays that support these processes; the overlays can provide (1) *reference structures* such as gridlines, (2) *highlights* such as outlines around important marks, (3) *redundant encodings* such as numerical data labels, (4) *summary statistics* such as the mean or max and (5) *annotations* such as descriptive text for context. We then present an automated system that applies user-chosen graphical overlays to existing chart bitmaps. Our approach is based on the insight that generating most of these graphical overlays only requires knowing the properties of the visual marks and axes that encode the data, but does not require access to the underlying data values. Thus, our system analyzes the chart bitmap to extract only the properties necessary to generate the desired overlay. We also discuss techniques for generating interactive overlays that provide additional controls to viewers. We demonstrate several examples of each overlay type for bar, pie and line charts.

Finally, we consider the broader context in which charts exist. News articles, reports, blog posts and academic papers often include graphical charts that serve to visually reinforce arguments presented in the text. Yet, connecting the text with the corresponding parts of a chart can be challenging, especially if the chart depicts many data points, or if the text paraphrases values depicted in the chart. To help readers better understand the relation between the text and the chart, we present a crowdsourcing pipeline to extract the references between them. Specifically, we give crowd workers paragraph-chart pairs and ask them to select text phrases as well as the corresponding visual marks in the chart. We then ask other workers to vote on the correctness of these references. Finally, we apply automated clustering and merging techniques to unify the references generated by multiple workers into a single set. Comparing the crowdsourced references to a set of gold standard references using a distance measure based on the F_1 score, we find that the average distance between the raw set of references produced by a single worker and the gold standard is 0.54 (out of a max of 1.0). When we apply our clustering and merging techniques the average distance between the unified set of references and the gold standard reduces to 0.37; an improvement of 32%. At the sentence level, we find our clustering and merging techniques produce an improvement of 43% over the average worker. Finally, we present an interactive document viewing application that uses the extracted references; readers can select phrases in the text and the system highlights the related marks in the chart.

To my parents and Wafa.

Contents

List of Figures

List of Tables

Acknowledgments

First, I'd like to thank my advisor, Maneesh Agrawala, for being an invaluable critical eye and making sure I never went down rabbit holes. He has inspired me to strive for perfection – although it is an unattainable goal, I know my work is better for it. Thanks also to my committee – Björn Hartmann, Marti Hearst, and Joe Hellerstein – for their invaluable advice and support.

Thanks are due to my former colleagues from the Dynamic Graphics Project at the University of Toronto, who put me on the path to a PhD in the first place. Thanks to Ravin Balakrishnan, who decided to answer a freshman's cold e-mail, and to Tovi Grossman, who patiently supervised me over the next three summers.

I have had the incredible fortune to collaborate with many others during the course of my studies; without their help, I would not be writing this document today. Thanks to Steven Drucker and Gonzalo Ramos at Microsoft; Gregorio Convertino, Ed Chi, Lichan Hong, Ben Hanrahan, and Bongshin Lee at PARC; and finally George Fitzmaurice and Tovi Grossman (again!) at Autodesk Research. It was a joy to collaborate with each and every one of you.

I also want to thank Jeffery Heer in particular, and his colleagues at Stanford (now UW) – Manolis Savva, Arti Chhajta, and Li Fei-Fei. I have had the great pleasure to work with Jeff Heer at both Berkeley and Stanford. As a model grad student and star junior faculty, he was what every grad student aspired to be. He was a second adviser to me when I was just starting grad school, and continued to be a patient and astonishingly insightful collaborator in the following years. His brilliance continues to be an inspiration to me, and I can't wait to see what he comes up with in the future!

The grad school journey would have been impossible without the support of my friends and colleagues ot Soda and BiD – Kenrick Kin, Andy Carle, David Sun, Kenghao Chang, Mark Fuge, Shiry Ginosar, Valkyrie Savage, Lora Oehlberg, Pablo Paredes, and Steve Rubin. Extra special thanks go to two of my colleagues and friends: Wes Willett and Anuj Tewari. Wes was the senior grad student I hoped to be – incredibly well-rounded, affable, and a blast to be around. Anuj Tewari, my deskmate for almost the entirety of my graduate career, helped keep things in perspective with his good cheer and boundless optimism.

Finally, the most thanks are reserved for my family. My parents instilled a work ethic and committment to education that has served me well to today. My older sister Janice blazed the post-secondary education trail, and my younger sister Melanie knows how to balance life and work. Last, but far from least, I owe the utmost thanks to my wonderful better half,

Wafa. Without your boundless patience, support, and love, I would not be who I am today. I can't wait to experience life and grow old together.

This dissertation was typeset using the ucastrothesis LaTeX template.

Chapter 1

Introduction

In the late 18th to early 19th centuries, William Playfair produced the first bar, pie, and line charts [90], ushering in the current era of statistical graphics. Since then, charts, graphs, and other visual depictions of data have become a primary vehicle for communicating quantitative information [109]. Bar, pie, and line charts, largely unchanged from Playfair's original designs, have become ubiquitous. Map-based visualizations are widely used to display political or demographic data. Designers and researchers have expanded the visualization repertoire with novel techniques for specialized forms of data, such as treemaps for hierarchical data [85], parallel coordinates for multidimensional data, and node-link diagrams for networks. Finally, the growth of aids for authoring visualizations, from libraries such as D3 [12] and Google Charts [39] to plug-and-play tools such as Tableau Public [104], have made the creation of visualizations easier than ever before.

However, while it is becoming easier to author visualizations, creating *effective* visualizations is still a challenge. A visualization author must make many design decisions, each of which could affect the manner in which a reader interprets the data, or how easy it is to decode the visualization (i.e., the *graphical perception* of the chart [25]). To explore the space of these decisions, we can use Card et al.'s visualization pipeline [17]. The author must first select a subset of the raw data to visualize, which results in a data table. The author must then apply visual encodings (such as position, color, shape, etc.) to the variables in the data table to produce a visual display. Finally, the author must choose what interactive controls, if any, to allow the viewer to customize the visualization by changing the slice of data, the visual mapping, or the view.

What the visualization reference model tells us is twofold: a visualization is the result of a complex set of decisions made by the author, and these decisions must take into account the tasks a viewer wishes to perform with the data. Thus, there is no ideal display of a set of data, since the designer cannot know the viewer's task *a priori*. Rather, there is a tradeoff between how well the visualization supports a specific task, and how well it supports a variety of tasks.

As an example, consider the two visualizations in Figure 3.1. Both visualizations display the same data: the number of songs played over a week by the author, organized by genre.

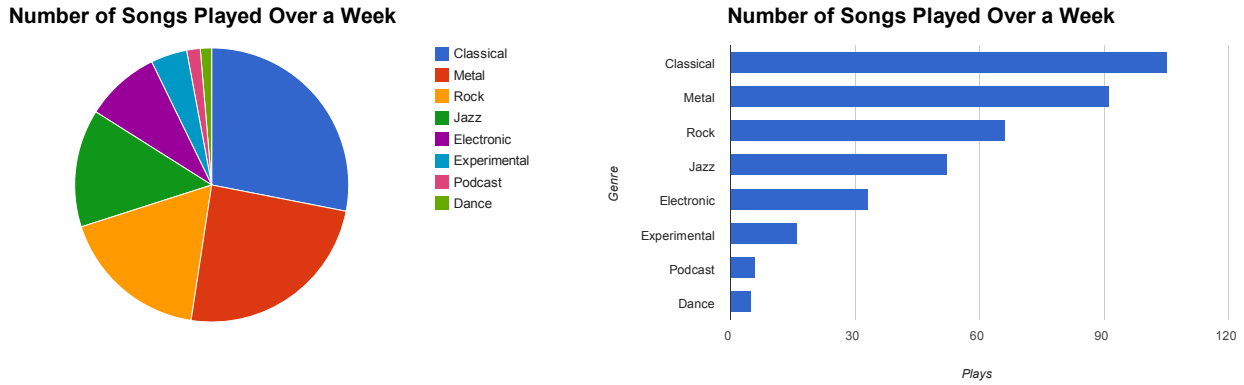


Figure 1.1: Two visualizations of the same data. Left: A pie chart showing the number of songs listened to over a week by the author, organized by genre. The chart makes it easy to identify the percentage of the songs were in the classical genre, but difficult to compare individual values. Right: Plotting the data as a bar chart enables more accurate comparisons of data values, but less accurate comparisons for part-to-whole relations [25; 71].

The left visualization displays the data as a pie chart, which makes it easy to infer part-to-whole relations (e.g., percentages) [25; 71]. For example, it is easy to see that classical music comprises about 28% of the total songs played, and that jazz and electronic tracks comprise around 23% of the total. However, pie charts are not the best visual representation for the viewer who wants to compare individual data values. It is difficult to tell whether more podcasts or dance tracks were played from the pie chart.

We can also display the same data set with a bar chart (Figure 3.1). This display allows us to compare individual data values much more easily. For example, the bar chart makes it clear that there are more podcasts than dance tracks. However, extracting proportions from the bar chart is much more difficult than it is from the pie chart. The viewer must mentally combine all of the bars to infer the total number of songs, then estimate the proportion the classical bar is of the total. The process is even more involved if it involves multiple genres, as in determining what percentage jazz and electronic tracks are of the total.

Despite the variety of design decisions necessary to produce a visualization and the virtually unlimited space of tasks users may have, all hope is not lost. In recent decades, researchers have made great progress in developing empirically grounded design principles for effective visualization [25; 33; 44; 71; 72; 87; 109], and in identifying the perceptual and cognitive processes that viewers undergo when reading a visualization [78; 87]. Furthermore, others have developed interactive techniques that expand the array of tasks a visualization can effectively support [36; 81; 86]. Taken together, these strands of research provide invaluable guidance to visualization designers.

While these advances in our understanding of effective visualization have had a significant impact on visualization design, many published charts exhibit poor design choices that hamper understanding of the underlying data and lead to unaesthetic displays. Some may

have been created before the availability of guidelines, while others may have simply disregarded them. In addition, even well-designed charts cannot be an ideal display for every task viewers might wish to perform.

Another important consideration for visualization design is to consider the context in which a visualization is used. Visualizations are rarely seen in isolation; rather, they are frequently paired with accompanying text, such as an academic paper or news report, to help support or explicate an argument. A full understanding of the chart may require finding the connections (or *references*) in the text to the chart, and vice versa. However, the references between the text and the chart can be ambiguous, obscure, or otherwise difficult to find, especially if the chart contains much more data than is referenced in the text.

This thesis explores methods for improving the usability of visualizations through three applications:

1. *Redesigning* existing visualizations;
2. *Overlaying* graphical elements that support the perceptual and cognitive processes users undergo when reading a visualization; and
3. Explicitly *displaying references* between the chart and surrounding text.

These applications require information about the chart beyond the raw pixels – in particular, we require the location of the marks and the underlying data table from the chart for all of our applications, and we require the references between the chart and surrounding text for the third application. In some cases, this information may be easily attainable. Perhaps the author already possesses the data table, or the chart has been dynamically produced with a visualization toolkit (such as D3 [12]) that exposes this information. However, the vast majority of published visualizations are in a raster format, which makes extracting the underlying data challenging. We thus present automated algorithms for extracting the marks and data from existing raster charts. Finally, we present a crowdsourcing pipeline for extracting the references between a chart and related text.

The structure of the thesis is as follows. We first survey the prior work on extracting data from and augmenting charts. We present techniques for extracting the marks and the underlying data from existing bar and pie charts and show how we can use this information to redesign existing visualizations. We then illustrate how we can use the marks and data from charts to produce graphical overlays, which help support the cognitive and perceptual processes a viewer performs when reading a chart. We then describe how we can extract the references between marks in a chart and related text via a crowdsourcing pipeline and clustering and merging algorithms, allowing readers to easily make the connection between the chart and the text. Finally, we conclude by describing avenues for future work. This includes improving and extending our mark and data extraction algorithms, and other applications that can leverage the marks and data extracted from existing charts.

1.1 Extracting Marks and Data from Charts

In Chapter 3, we describe algorithms for extracting marks and data from bitmap visualizations. These algorithms are part of the larger *ReVision* system, which also identifies the type of chart and produces alternate redesigns of the input chart using the extracted marks and data and design guidelines. ReVision also supports stylistic redesign; users can change mark types, colors or fonts to adhere to a specific visual aesthetic or brand.

We present a set of chart-specific techniques for extracting the graphical marks and the data values they represent from a visualization. Our implementation focuses on bar and pie charts. We first locate the bars and pie slices that encode the data. We then apply heuristics to associate the marks with related elements such as axes and labels. Finally, we use this information to extract the table of data values underlying the visualization. We exclude any bar and pie chart images with 3D effects or non-solid shading. We achieve an average accuracy of 71.4% in extracting the marks and an average accuracy of 67.1% in extracting the data after we successfully extract the marks. Finally, we demonstrate how these contributions can be combined with existing design guidelines to automatically generate redesigned visualizations.

1.2 Augmenting Charts to Support Perceptual Processes

Reading a visualization involves a combination of perceptual and cognitive processes. For example, to extract the value encoded by a bar in a bar chart, the viewer must find the relevant bar, mentally project the top of the bar to a point on the y-axis, and infer its value using the labeled tick marks on the axis. While other types of charts (e.g., pie charts, line charts, scatterplots, etc.) use different visual encodings for the data, common chart reading tasks such as extracting, comparing and aggregating values usually involve a similar sequence of perceptual and cognitive processes [87; 61; 69; 78]. However, many visualizations published in newspapers, reports, books and on the Web only support a subset of these processes.

In Chapter 4, we use the extracted marks and data from ReVision to produce *graphical overlays*—visual elements that are layered onto a chart to facilitate a larger set of perceptual and cognitive processes involved in chart reading. We identify five main types of overlays, each designed to support different processes. (1) *Reference structure* overlays, such as gridlines (Figure 4.1-2), aid the viewer in extracting and comparing values. (2) *Highlight* overlays draw the viewer's attention to certain marks by creating perceptual groups of marks. (3) *Redundant encoding* overlays allow viewers to extract numerical data in multiple ways and can be used to better depict some aspects of the data such as trends (Figure 4.1-3). (4) *Summary statistics* overlays depict aggregate information about the data set such as its mean, median or maximum (Figure 4.1-4). (5) *Annotation* overlays help viewers communicate and collaboratively analyze charts. While all five overlay types add visual elements to aid chart reading, they can also increase visual clutter. As a result, many published visualizations

include few, if any, such overlays.

We then present an automated system that applies user-chosen graphical overlays to existing chart bitmaps. By targeting existing charts, our system allow users to tailor published visualizations to better support the chart reading tasks they wish to complete. Our system takes a bitmap chart, an overlay type, and an optional set of user-specified parameters as inputs and outputs a graphical overlay for the chart. Our approach is based on the key insight that generating most graphical overlays requires only knowing the properties of the visual marks and axes that encode the data and does not require access to the underlying data values. Thus, our system analyzes the chart bitmap using ReVision (for pie and bar charts) and DataThief [111] (for line charts) to extract only the properties necessary to generate the desired overlay. We also show how our system can be used to generate interactive overlays and how it can place overlays underneath the marks in a chart.

1.3 Associating Marks in Charts with Surrounding Text

Charts are most often embedded in a document, such as a news article, report, blog post, or an academic paper. They can provide additional data not mentioned in the text to give readers more context and allow them to make their own inferences. Thus, for readers to fully understand such a document they must parse all of the *references* between the text and the corresponding visual marks (e.g. bars, lines, points, pie slices, etc.) in the charts.

However, identifying such references between the text and the chart can be challenging. Often, the text only refers to a subset of the data in the chart and the reader must perform complex visual comparisons to identify the correct subset. In other cases, the text may paraphrase values in the chart and require the reader to bring external information to bear. For example, the text may use the term "EU" to refer to a subset of the European countries in the chart.

In Chapter 5, we present a crowdsourcing pipeline that takes a document containing text and one or more charts as input, and extracts the references between the text and the chart. In the preprocessing stage of our pipeline we use a mix of manual techniques and ReVision (Chapter 3) to segment the document into paragraph-chart pairs and then extract the marks and data from each chart. In the reference extraction stage, we give crowd workers paragraph-chart pairs and ask them to select text phrases as well as the corresponding visual marks in the chart. We then ask another set of crowd workers to vote on the correctness of each extracted reference. Finally, we apply automated clustering and merging techniques to unify the references generated by multiple workers into a single set of references.

We compare the crowdsourced references to a set of gold standard references created by two experts using a distance measure based on the F_1 score [21]. We find that the average distance between the raw set of references produced by a single worker and the corresponding gold standard references is 0.54 (out of a max of 1.0). When we filter the references using the crowdsourced votes and apply our clustering and merging techniques, the average distance between the unified set of references and the gold standard reduces to 0.37; an improvement

of 32%.

Finally, we present an interactive document viewer that uses the extracted references to improve the reading experience by augmenting the chart and the text. Readers can select phrases in the text and our application automatically highlights the related marks in the chart. This application demonstrates how access to the references can help readers better understand the relationship between the text and charts in a document.

Chapter 2

Related Work

This thesis focuses on improving the usability of existing charts by (a) extracting the marks, data, and references to text from existing charts, and (b) building applications with this data that improve usability of charts for a variety of users and tasks. We build upon prior work on algorithmic techniques to extract the marks and data from charts, and we take inspiration from crowdsourcing systems for our reference extraction pipeline. Finally, we build on research into graphical perception and theory on the perceptual and cognitive processes viewers undergo when reading charts in order to build our applications.

2.1 Extracting Marks from Charts

Charts are easily parseable by humans, but not by machines. However, recovering the underlying data from a chart would be very useful for a variety of applications, such as improving information retrieval engines. In our case, we want to recover the underlying data in order to modify or enhance the chart. To this end, some researchers have investigated algorithms for automatically extracting the graphical marks from raster charts. Although approaches vary, prior methods first extract edges or boundaries in the chart, then apply rules to combine the edges into semantic units, such as axes or bars.

Zhou and Tan [121] present algorithms that identify bar charts in a monochrome document and extract the bars from the bar charts. They first separate a document into text and graphics by finding connected components and classifying large components as graphical elements. They then use boundary tracing and the Hough transform to identify bars. Yokokura and Watanabe [119] suggest an alternate solution to the problem of extracting the graphical marks in bar charts. They use histograms to identify straight lines in the chart, then apply rules to extract the bars and axes.

Huang et al. [50; 51] and Liu et al. [68] extend the rule-based approach to include pie, line, and low-high charts in addition to bar charts. They generate edge maps, vectorize the edge maps, and use rules to identify the chart type and extract the graphical marks. Yang et al. [118] extended this work by building a system to allow humans to correct automatically generated vector representations of charts.

In these prior systems, researchers focused on extracting the mark geometry rather than the underlying data. In addition, their techniques rely on an accurate edge map, which can be difficult to retrieve from real-world charts, due to large variance in image quality. In Chapter 3, we build on this prior work by describing robust algorithms that extract both the marks and the data from bar and pie charts.

2.2 Crowdsourcing in Online Labor Markets

For some problems, state-of-the-art algorithms do not yet give acceptable results. In Chapter 5, we tackle one such problem: extracting references between text and charts. In these cases, online labor markets, such as Amazon Mechanical Turk[1], offer a compelling alternative to potentially brittle automatic algorithms for data extraction tasks. These markets, in which workers are paid small fees to perform short tasks called microtasks, are already widely commercially used for tasks such as translation or data transcription, through such companies as CrowdFlower [30], MobileWorks [74], and Captricity [16]. They can provide robust, accurate results at a fraction of the cost of expert annotators or workers.

Researchers from a variety of fields have used crowdsourcing to complete small tasks, such as gathering responses to experimental stimuli or annotating image data. Others have investigated how complex tasks can be decomposed into simpler tasks amenable to crowdsourcing. In Chapter 5, we describe a crowdsourcing pipeline for extracting references between text and charts in which we build upon both strands of prior work in the design of our pipeline.

2.2.1 Crowdsourcing Perception Experiments

In the context of visualization, crowdsourcing has become a primary means to recruit participants for experiments, as it is a cheaper and quicker alternative to standard laboratory experiments. Heer and Bostock [44] replicated classic graphical perception results on Mechanical Turk, showing that basic perception experiments are robust enough to conduct on microtask platforms despite the lack of controls. Since then, many others have used Mechanical Turk to conduct visualization experiments, including additional studies of graphical perception [38; 60], and studies of the effect of social information [52] and metaphor [122] on quantitative judgments in charts.

2.2.2 Crowdsourcing NLP Tasks

Our work on extracting references in text to charts also draws on related work on crowdsourcing in the natural language processing (NLP) community. Snow et al. [88] demonstrated the viability of crowdsourcing to cheaply generate large sets of labeled training examples for a variety of text analysis tasks, including affect recognition, word similarity, recognizing textual entailment, temporal ordering, and word sense disambiguation. Others have used

[1] http://mturk.com

it to create training data for sentiment analysis [49], identify entities across languages [73], and find paraphrases for noun compounds [75]. Callison-Burch and Dredze [15] provide a recent survey of uses of crowdsourcing in NLP research. We use crowdsourcing to address a related, but novel problem: extracting references between the text of a document and the charts within it.

2.2.3 Building Crowdsourcing Pipelines

Some tasks, such as responding to stimuli in a graphical perception experiment, or identifying objects in an image, can easily be completed in a single microtask. However, other tasks, such as editing a paper or planning a vacation, may require multiple steps. There has been much recent work on designing crowdsourcing pipelines for such complex tasks. The primary challenges here are (a) to decide how to split the task into smaller microtasks that can be completed by the workers in parallel, and (b) to develop techniques for combining the resulting work into a complete solution for the original task. Recent examples of crowdsourcing pipelines for complex tasks include crowdsourcing taxonomy creation [20], explaining outliers and trends in data analysis [115; 116], and generating answers to uncommon queries [7]. Others have designed toolkits such as TurKit [67], CrowdForge [57], Turkomatic [63], and Jabberwocky [1], to help developers implement complex crowdsourced pipelines. Our crowdsourcing pipeline for extracting references from text to charts draws inspiration from the design of these other pipelines and toolkits.

2.3 Perceptual and Cognitive Models for Visualization

Researchers have developed a variety of models of the perceptual and cognitive processes viewers perform as they comprehend a graph [18; 62; 69; 78; 82]. All of these models share a similar high-level structure; the viewer first perceives the visual elements of the chart, associates the visual elements with graphical concepts (e.g., recognizing a bar, or set of bars), and finally extracts the desired information from the chart. This process may iterate multiple times. For example, in a line chart with multiple intersecting lines, viewers may first process each line individually, then combine that information to complete their task. In Chapter 4, we describe a taxonomy of graphical overlays that are designed to support the early stages of this process: the perception of visual elements and the association of elements with graphical concepts. Our taxonomy draws on the wealth of prior research in these areas.

2.3.1 Graphical Perception

Many researchers have investigated how visual encodings such as length, area, color, and texture affect graph comprehension. At a basic level, visualizations help us identify like and non-like elements, perceive rank-order relations, and compare quantities. Both theoretical and empirical evaluations assess how different visual encoding techniques facilitate these basic judgments, typically expressed in terms of the speed and accuracy of decoding.

Bertin [8] was among the first to consider this issue. Based on his experience as a cartographer, he proposed an ordering of visual encodings for three common types of data: nominal (or categorical), ordinal, and quantitative. He wrote that spatial encodings are superior to other encodings, and that hue effectively encodes nominal data but not quantitative data.

S. S. Stevens [94; 95] drew on psychological research to model the mapping between the physical intensity of a stimulus (e.g., a shape's length or area) and its perceived intensity as a power law: $P = kI^{\alpha}$, where P is the perceived intensity, I is the physical intensity, k is an empirically determined scale constant, and α is the power law exponent. If $\alpha > 1$, perception tends toward overestimation: e.g., doubling the weight of an object makes it feel more than twice as heavy. If $\alpha < 1$, perception tends towards underestimation: e.g., an object twice as large may not look so. If $\alpha = 1$, there is no systematic bias to the perceived intensity. Stevens investigated mostly non-visual stimuli (e.g., smell, weight, and taste), but did report results for visual length and area. He found the exponent for length to be $\alpha \approx 1$ and the exponent for area to be $\alpha \approx 0.7$ [96], suggesting that people tend to accurately estimate length differences but underestimate area differences. Others have replicated this finding, albeit with varying exponent values [24; 89; 106].

Cleveland & McGill [25; 26] extended Bertin's and Stevens' work by applying results from psychology to provide empirical grounding for the order of visual encodings. Their human-subjects experiments established a significant accuracy advantage for position judgments over both length and angle judgments. Since then, others have made great progress in furthering our knowledge of the how visual encodings and their interactions affect graphical perception. Researchers have studied color [46; 66; 100], shape and aspect ratio [38; 44; 60], and data-dense displays such as horizon graphs [45], braided time series [55], and treemaps [60; 105]. We draw upon this work in our redesign application in Chapter 3 and in the design of our overlays in Chapter 4.

2.3.2 Perceptual Processes

Simkin and Hastie [87] describe the elementary perceptual processes that viewers use when extracting and comparing values, including anchoring, scanning, projection, superimposition, and detection (Figure 2.1). We designed our overlays to facilitate these elementary processes.

Anchoring occurs when the viewer mentally segments a mark into regular divisions. The segmentation is based on the encoding type – a bar may be divided in half, as in Figure 2.1-a, while an entire circular pie might be divided into quarters. Scanning occurs when the viewer mentally interpolates from an anchor to the mark's encoded value. In Figure 2.1-b, the viewer scans from the anchor down to the value of the lower gray bar. Projection occurs when the viewer mentally draws a horizontal or vertical line to facilitate comparison of values. In Figure 2.1-c, the viewer projects the top of the left bar to the right bar to compare their heights. Superimposition occurs when the user mentally moves a mark near another mark. In Figure 2.1-d, the viewer superimposes the right dark gray bar onto the unfilled portion of the left bar to make an aggregate judgment. Finally, detection is a quick

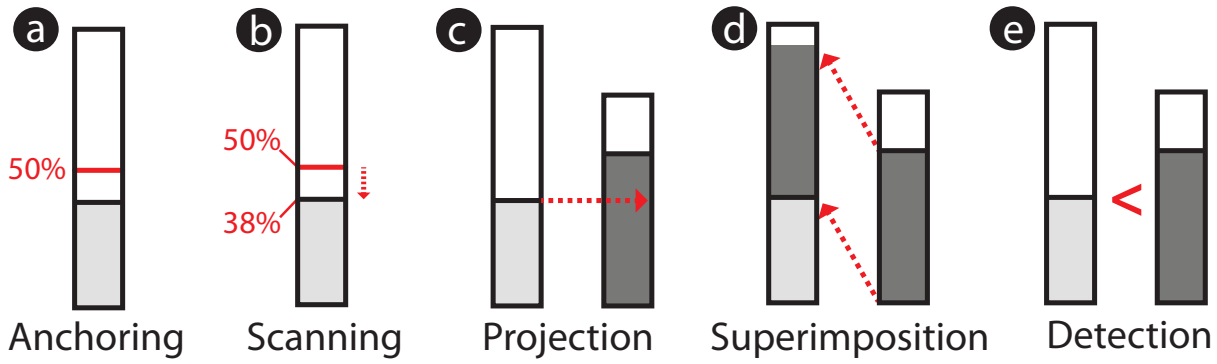


Figure 2.1: Simkin and Hastie's [87] elementary processes. (a) Anchoring occurs when a viewer segments a mark to extract a value. (b) Scanning occurs when a viewer estimates a mark's value based on the anchor. (c) Projection occurs when a viewer extends a horizontal or vertical line. (d) Superimposition occurs when a viewer mentally moves a mark's location. (e) Detection occurs when a viewer decides which of two marks is larger or smaller.

process in which the viewer compares the relative magnitudes of two marks. In Figure 2.1-e, the viewer can quickly tell that the right, dark gray bar is larger than the left, light gray bar.

2.3.3 Cognitive Models

Researchers have also created models that describe how a viewer mentally breaks a chart into its components and interprets the meaning of each component. In particular, a number of researchers have suggested that viewers use *graph schemas*, which are learned cognitive structures that describe the components of charts of different types [61; 69; 78]. For example, a schema for a bar chart might contain axes and bars. Viewers instantiate a graph schema based on the visual elements they perceive and use the schema to query the chart. If the viewer's query cannot be answered by retrieving a component in the schema, the viewer may need to use additional cognitive processes (e.g., aggregation) to answer the query. Our overlays provide additional visual elements that can extend viewers' graph schemas. For example the overlay might provide numerical data labels for each mark or depict the mean value of the data. By extending the graph schema and providing direct access to additional information, such overlays reduce cognitive load.

2.4 Automated Visualization Design

The body of graphical perception research has also produced many design guidelines for effective visualization. Subsequently, researchers have used these guidelines to automatically produce effective visualizations. Mackinlay's APT [71], one of the earliest such systems,

generates charts guided by rankings of visual variables for specific data types such as nominal, ordinal, or quantitative data. Stolte et al.'s Polaris [97] is a system for generating small multiples [109] displays based on user queries of a multidimensional data table. Mackinlay et al. [72] extend this work to support a range of automated visualization designs. Such research has resulted in commercial systems, including Tableau [103] and Spotfire [92]. These systems assist users in producing visualizations directly from data. Our work on automatic redesign (Chapter 3) is complementary to these earlier systems: once we have extracted a data table, we could feed it into any of these systems to generate improved alternative designs.

2.5 Layering in Visualizations

Layering information and visual elements is a common approach to designing visualizations. For example, William Playfair's economic charts from the 18^{th} century include gridlines layered onto line charts [79]. Gridlines are especially useful for extracting values from charts and commonly appear in visualizations published today. Other visual elements that are frequently layered onto visualizations include labels and highlights. Visualizations published on the Web sometimes provide interactive overlays such as the mouse-over data tooltips in Google Finance charts [40] and drop-down guidelines that follow the mouse cursor in the Wall Street Journal's Foursquare visualization [102]. Web-based maps often layer additional information such as the crime incidents in Stamen's Crimespotting [70] or building heat consumption in the UK government's National Heat Map [112]. In the academic literature, researchers have developed specialized graphical overlays for specific tasks, including illustrating links between data tuples in a treemap [32]; illustrating links between multiple visual entities, such as text, images, or marks on a chart [93]; illustrating set relations [28]; providing references to offscreen items [5; 42]; and providing annotations [3; 47; 58]. In all of these examples, both published and academic, each overlay is designed for a specific application and is created based on knowledge of the data underlying the visualization. As a result these overlays cannot be applied to other existing charts. In Chapter 4, we present a general-purpose tool for applying graphical overlays to existing chart bitmaps.

Although layers are commonly used in visualizations today, Tufte was amongst the first to investigate the use of "layering and separation" in visualization design as a general principle [110]. He presents a number of visualizations that carefully use color, size and other retinal variables to visually distinguish different visual layers. For example, he explains how a musical staff composed of thin gray gridlines rather than thicker black lines allows viewers to visually separate the staff from the notes. Such layering allows viewers to easily direct their attention to the layer of their choice. Recently researchers have begun investigating how layered gridlines should be designed to facilitate such visual separation. They have conducted experiments to determining the minimum discriminable alpha value for layered gridlines [4; 99] and the optimal distance between gridlines [44]. We apply some of these results in the design of our overlays.

2.6 Visualization Annotation

Visualization researchers have recognized that the most effective charts use labels, captions and other text annotations to clarify and accentuate important points [23]. Such text annotations can direct the readers attention through a visualization [84], provide additional semantics and meaning for the data [14], and emphasize specific interpretations [53]. Researchers have also explored how annotations are used in the context of collaborative visual data analysis [47; 117]. We include annotations as one of the categories in our graphical overlay taxonomy (Chapter 4).

Researchers have begun to explore techniques for automatically adding text annotations to a chart for specific tasks, such as to draw attention to outliers and trends in the data [56] or to provide additional context for stock chart data [54]. These techniques assume that the text used for annotation resides in the dataset (e.g., as metadata) or can be retrieved through domain-specific searches (e.g., by searching a news database for stock ticker symbols). Our crowdsourcing pipeline for extracting references between text and charts (Chapter 5) is premised on the idea that charts often appear within documents that contain additional text explaining the chart. Our interactive document viewing application is designed to highlight such references and thereby annotate the chart with explanatory text.

Chapter 3

Extracting Marks and Data from Charts for Redesign

Despite great advances in our understanding of how to create effective visualizations, many current visual displays do not follow best-practice design guidelines. For example, consider the pie chart in Figure 3.1 (left), which depicts data concerning the 2005 research budget of the National Institutes of Health (NIH). The design of this chart could be improved in multiple ways: slices are ordered haphazardly, labels are placed erratically, and label text competes with saturated background colors. Moreover, the chart encodes values as angular extents, which are known to result in less accurate value comparisons than position encodings [25; 44; 87]. Figure 3.1 (right) shows the same data in a redesigned visualization: the bar chart sorts the data, uses a perceptually-motivated color palette [98], and applies a position encoding.

For analysts working with their own data, automated design methods [71; 72] based on visual design principles [25; 33; 44; 109] can lead to more effective visualizations. However, the vast majority of visualizations are only available as bitmap images. Without access to the underlying data it is prohibitively difficult to create alternative visual representations.

In this chapter, we present *ReVision*, a system that takes bitmap images of charts as input and automatically generates redesigned visualizations as output. For example, ReVision produces Figure 3.1 (right) as a suggested redesign when given Figure 3.1 (left) as input. ReVision identifies the type of chart, extracts the marks (visual elements that encode data) and underlying data, and then uses this information in tandem with a list of guidelines to provide alternate designs. ReVision also supports stylistic redesign; users can change mark types, colors or fonts to adhere to a specific visual aesthetic or brand.

Portions of this chapter previously published by Manolis Savva, the author (as co-first author), Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer in [83].

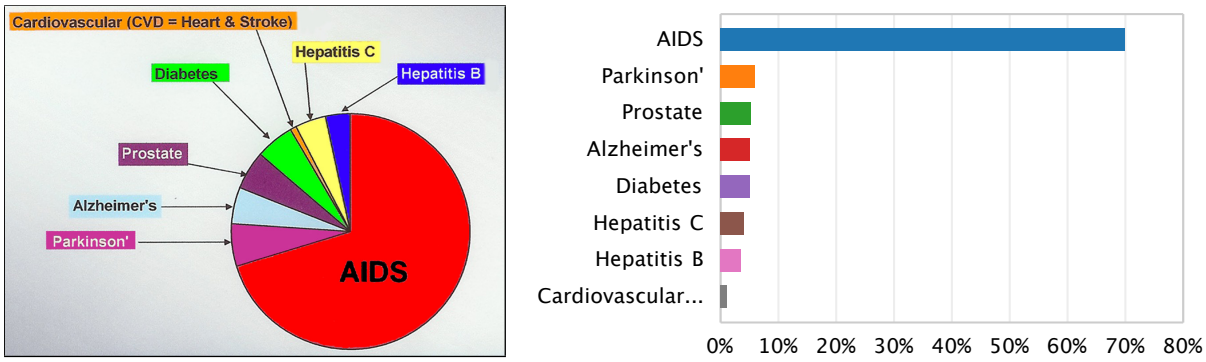


Figure 3.1: Chart redesign. Left: A pie chart of NIH expenses per condition-related death. The chart suffers from random sorting, highly saturated colors, and erratic label placement. Right: Plotting the data as a sorted bar chart enables more accurate comparisons of data values [25; 71].

3.1 System Overview

ReVision is comprised of a three stage pipeline: (1) chart classification, (2) mark and data extraction, and (3) redesign. In stage 1, ReVision classifies an input image according to its chart type. This stage uses a corpus of test images to learn distinctive image features and train classifiers. In stage 2, ReVision locates graphical marks, associates them with text labels, and extracts a data table. In stage 3, ReVision uses the extracted data to generate a gallery of alternative designs. In this chapter, we will focus on stages 2 and 3 of ReVision. We briefly describe the classification stage, but refer the reader to the published paper for full details [83].

3.2 Stage 1: Classifying Chart Images

ReVision determines the type of chart using both low-level image features and extracted text-level features. We propose a novel set of features that achieve an average classification accuracy of 96%.

3.2.1 Extracting image features

To create our image features, we used the approach of Coates et al. [27]. They first create a "codebook" of representative image patches (small rectangular areas), then use the codebook to produce feature vectors for each image. We first normalized the charts to fit a 128×128 pixel square, preserving the aspect ratio by padding with the background color. We then randomly sample 100 6×6 pixel patches from each chart in the corpus. Next, we create a codebook of representative image patches by applying K-means clustering to the chart. We found $K = 200$ to work well with our test corpus.

To create the image feature vectors, we first extract a 6×6 patch centered on every pixel in the image, then find the nearest codebook patch for each of the extracted patches. This results in a 128^2 sized vector. To reduce the dimensionality, we divide the chart into four quadrants and produce a histogram of activated codebook patches for each quadrant. This results in a $4 \times 200 = 800$ element feature vector. Finally, we perform classification using Support Vector Machines (SVMs) [29] with a quadratic kernel function. We use the SVM implementation provided by the WEKA [37] framework.

3.2.2 Extracting text features

The position of text in a chart can provide information about the type of chart. For example, axis labels tend to lie along a line, whereas labels on a pie chart lie outside or inside each sector. We thus use the locations of the text in the chart to create a text feature vector, which we combine with the image feature vector in our classifier.

We designed a tagging interface to annotate chart images with the position, size, angular orientation and content of text regions (Figure 3.2). Our tool extracts the text image region and performs OCR using the Tesseract open source OCR engine[1]; the user can correct the OCR output if desired. While manual text annotation is tedious, it is amenable to crowdsourcing and we designed the tagging UI with this scenario in mind. Furthermore, connected components analysis or more sophisticated approaches [19] can extract text region bounding boxes automatically. These methods could allow fully unsupervised textual feature extraction from a visualization corpus.

To create the text features, we create a binary mask for each chart where a pixel is on if a text region contains it, off otherwise. We then divide the chart into 8×8 pixel regions and compute the proportion of text pixels in each region. We then linearize the values into a 64 element vector. We also append histograms of text region length, width, center position, and pairwise orientation and pairwise distance between text region centers to the text feature vector.

3.2.3 Classification results

We tested the results using a corpus published by Prasad et al. [80], who explored the same problem of classifying charts. Their corpus contains 667 charts drawn from five categories: bar charts (125), curve plots (121), pie charts (134), scatter plots (162) and 3D surface plots (125). Table 3.1 illustrates the results of our classifier when using image features, text features, and both features for multi-class SVMs and binary SVMs. When using binary SVMs, we achieve a classification accuracy of 96% on average. In the published paper, we describe experiments with a much larger corpus of 2,601 images, and find an identical 96% classification accuracy with binary SVMs [83].

[1] https://code.google.com/p/tesseract-ocr/

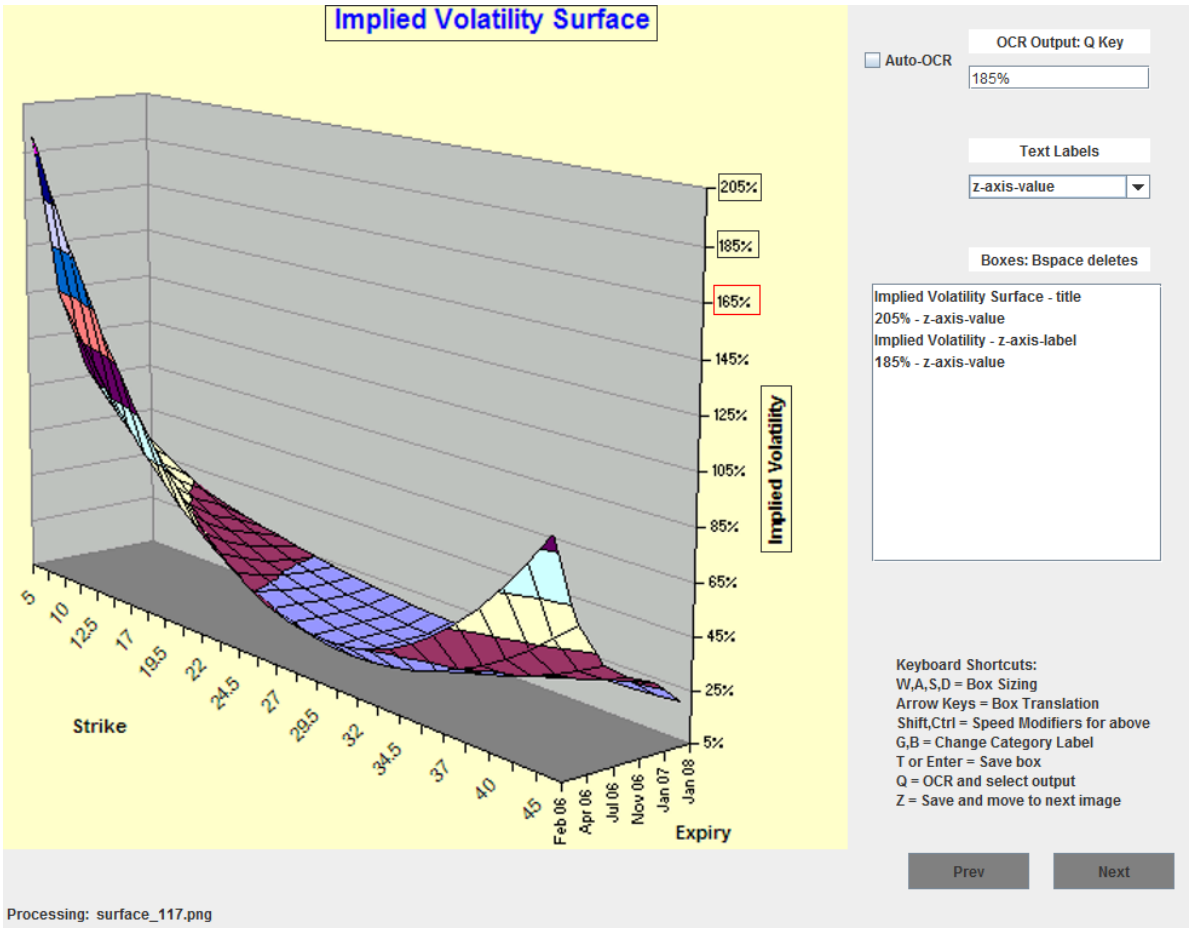


Figure 3.2: Our interface for tagging textual features of a chart image. The user positions and orients a bounding box for each block of text and then retrieves the text via OCR, potentially followed by hand-correction.

	Prasad [80]	Image	Text	All	Binary
Bar	90%	85%	57%	89%	95%
Curve	76%	75%	50%	83%	92%
Pie	83%	93%	84%	95%	97%
Scatter	86%	91%	64%	91%	97%
Surface	84%	90%	71%	94%	97%
Average	**84%**	**88%**	**66%**	**90%**	**96%**

Table 3.1: Classification accuracy for the corpus from Prasad et al. [80]. We compare Prasad et al. to our method using image features (first two columns), text features (1st and 3rd), and both (1st and 4th) in multi-class SVMs. The last column shows results for both features using binary SVMs.

3.3 Stage 2: Mark and Data Extraction

After categorizing charts by type, ReVision proceeds to locate graphical marks and extract data. Our implementation focuses on mark and data extraction for bar and pie charts, two of the most popular chart types. We first *preprocess* the input charts to remove noise and compression artifacts. Next, we perform *mark extraction*, which locates graphical marks such as bars and pie slices by fitting models of expected shapes to image regions. Finally, we perform *data extraction*, which infers the mapping between data space and image space, associates labels with marks, and then extracts a data tuple for each mark.

Our algorithms are based on a few simplifying assumptions:

- Charts contain 2D marks and do not contain 3D effects.
- Each mark is solidly shaded using a single color. Marks are not shaded using textures or steep gradients.
- Marks encode a data tuple, where at least one dimension is quantitative and one dimension is nominal. For example, in a bar chart the height of a bar represents a quantitative value, while a nominal label often appears below the bar.
- Bar charts do not contain stacked bars.
- Bar chart axes appear at the left and bottom of the chart.

Based on these assumptions we develop simple, robust data extraction algorithms that apply to a significant number of real-world charts. For example, 42% (52/125) of the bar and 43% (53/125) of the pie charts in the Prasad et al. corpus [80] follow these assumptions.

3.3.1 Preprocessing: Smoothing the images

We have found that chart images from the web are often heavily compressed and noisy, which can make it difficult to identify mark borders. To reduce such artifacts, we first apply

the bilateral filter [107] to each bar or pie chart image from the classifier. The bilateral filter smooths the image in both its domain and its range to remove small variations in color while retaining sharp edges in the image. It takes the form

$$h(\mathbf{x}) = k(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\mathbf{x}) c(\xi, \mathbf{x}) s(f(\xi), f(\mathbf{x})) d\xi \tag{3.1}$$

where

- $\mathbf{x}$ is the neighborhood center,
- $f(\mathbf{x})$ is the value of the pixel at $\mathbf{x}$ of the original image,
- $h(\mathbf{x})$ is the filter response at $\mathbf{x}$,
- $c(\xi, \mathbf{x})$ is a function that measures the closeness of ξ and $\mathbf{x}$ (i.e., the domain distance),
- $s(\xi, \mathbf{x})$ is a function that measures the similarity of ξ and $\mathbf{x}$ (i.e., the range distance), and
- $k(\mathbf{x})$ is a normalization constant.

We perform smoothing using Lanman's MATLAB implementation [64]. We first convert images to the LAB colorspace. We then use 2D Gaussian kernels as our closeness and similarity functions. We set the standard deviations to $\sigma_{closeness} = 2$ and $\sigma_{similarity} = 19.6$, noting that the image luminance has a range of $[0, 100]$ in the LAB colorspace. Finally, we restrict the smoothing functions to a neighborhood of 5x5 pixels. We found that these settings worked well with our corpus of charts.

3.3.2 Extracting Marks from Bar Charts

Next, we locate the marks. We describe our mark extraction procedures for bar and pie charts in turn.

Figure 3.3 shows the steps of our bar extraction algorithm. We identify bars by looking for rectangular connected components and then use the bar locations to extract the axes.

Find rectangular shapes

We first extract connected components from the filtered image by grouping adjacent pixels of similar color, specifically those with an L^2 norm less than 0.04 in normalized RGB space (Figure 3.3b). We then identify rectangular connected components by computing how much each component fills its bounding box. If component pixels fill more than 90% of the bounding box, we classify the component as a rectangle; otherwise, we discard the component (Figure 3.3c).

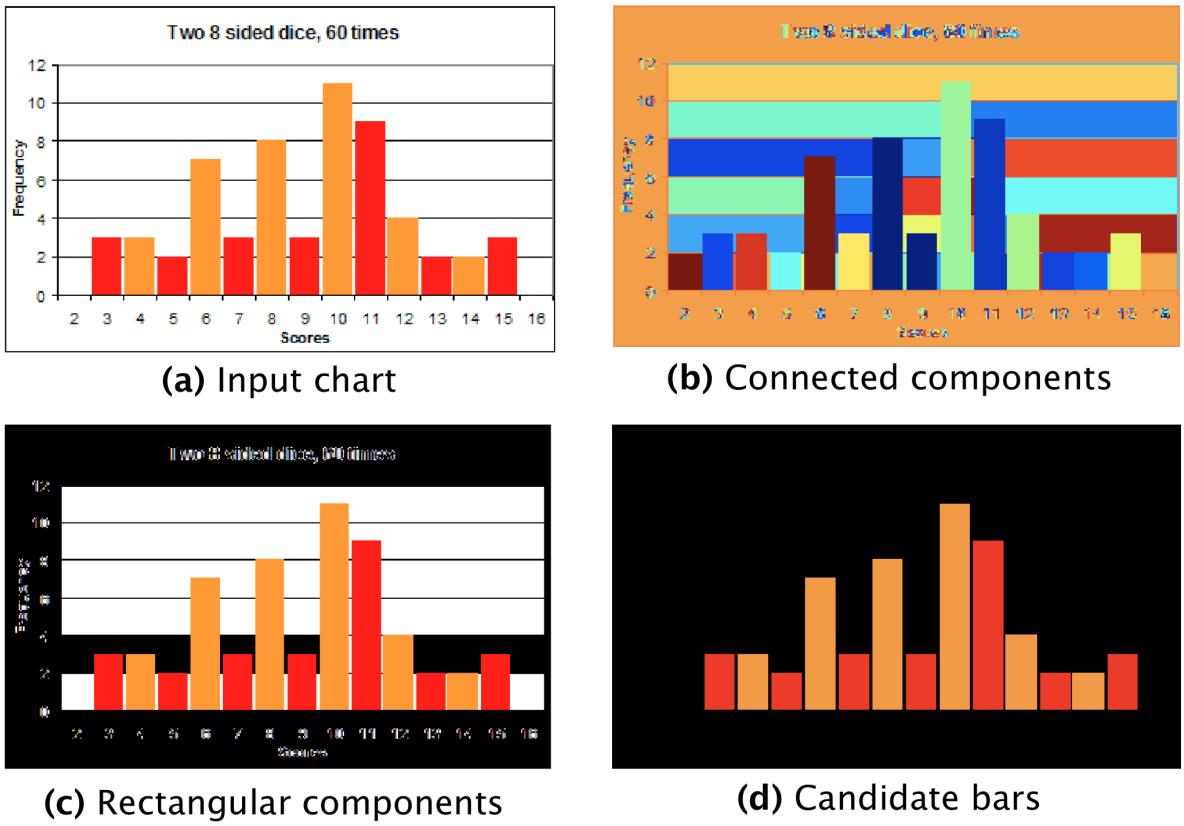


Figure 3.3: Bar extraction procedure. We compute connected components (shown false-colored) and discard non-rectangular components. We keep rectangles whose color differs from surrounding colors (see inset figure on this page), that touch the baseline x-axis, and whose height is greater than 2 pixels or whose width is near the average width of the other candidate bars.

Remove background rectangles

The remaining rectangles are likely to include all data-encoding bars, but may also include background rectangles formed by gridlines. In Figure 3.3c, the background rectangles are colored white, while the true bars are orange or red. Background rectangles are usually shaded with the same color as adjacent rectangles, since they adjoin other background rectangles. On the other hand, bars tend to be shaded with different colors than adjacent rectangles, due to bars being shaded differently or gaps between bars. We test for background rectangles by comparing each rectangle's color to the color of four points outside the rectangle, as shown in Figure 3.4. We choose outside points by first finding the color of pixels bordering the rectangle. We then move away from the rectangle until we find a pixel whose color differs from the border pixel, or is 5 pixels away from the rectangle. We then compare the color of the outside points to the average color of the interior of the rectangle by computing the L^2 norm in normalized RGB space. If any one of the outside points has a

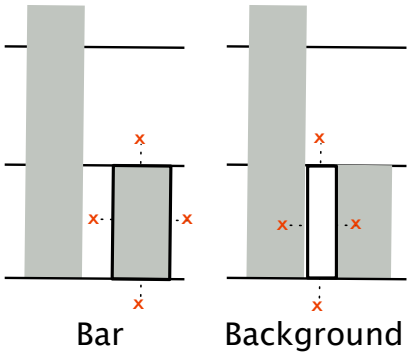


Figure 3.4: Identifying background rectangles. We compare the color of four points outside of a rectangle to the color inside the rectangle. If any of the colors of the outside points are sufficiently close to the color of the rectangle, we classify the rectangle as a bar. Otherwise, we classify it as a background rectangle.

color that is almost the same as the interior color (i.e., has an L^2 norm less than 0.01), we classify the rectangle as part of the background and discard it; otherwise, we classify it as a candidate bar.

Remove small rectangles

Finally, we also discard very small and thin rectangular components (with width or height less than 2 pixels), as such small components are unlikely to represent bars and are often due to image artifacts or text. However, some of these components do represent very small bars, and we add them back to our set of candidate bars later in the extraction procedure.

Infer chart orientation

Bar charts may be oriented horizontally or vertically. Since bars encode data using length, they vary in one dimension and remain fixed in the other dimension; e.g., vertical bars vary in height but maintain a fixed width. To identify which dimension varies most we build histograms of the widths and heights of the candidate bars and find the mode of each histogram (Figure 3.5b). The histogram with the strongest mode represents the constant dimension and gives us the orientation of the chart: vertical if the width histogram has the strongest mode, horizontal if the height histogram has the strongest mode. For brevity, we will describe our algorithms assuming a vertical bar chart, but our system works with horizontal bar charts using analogous analysis techniques.

Baseline axis extraction

For most vertical bar charts the baseline axis is the x-axis that touches the bottom of the bars. However, in some bar charts (e.g., those with negative values), the top of some bars

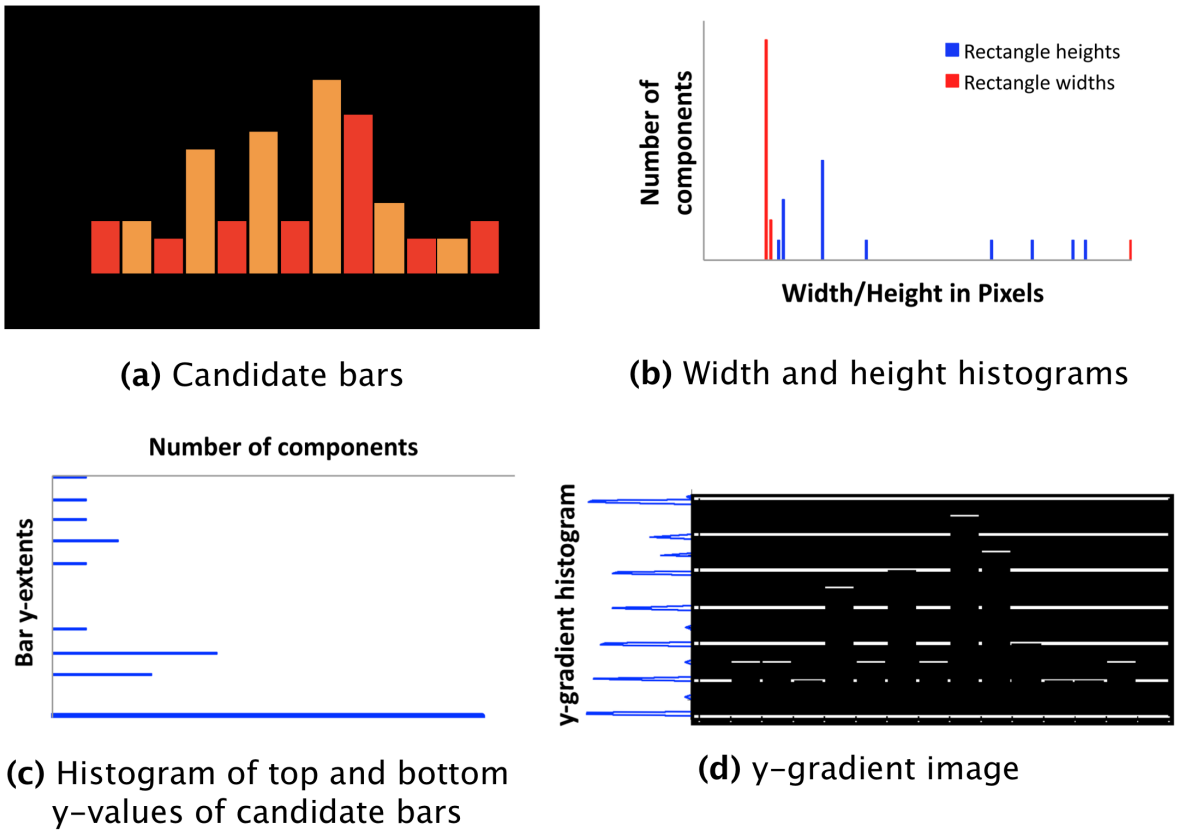


Figure 3.5: Inferring the orientation of the chart and locating the baseline axis. (a) Candidate bars. (b) Histograms of widths and heights of the bars. Note that the mode of the width histogram (red) is stronger than the mode of the height histogram (blue), indicating that the chart is vertical. (c) To estimate the baseline axis, we build a histogram of the top and bottom y-values of candidate bars and treat the mode of this histogram as our initial estimate of the baseline axis. (d) Many charts draw the baseline axis as a solid line. To refine the axis location, we compute the y-gradient image of the initial chart and then sum the rows to create the y-gradient histogram. We treat the peak of this histogram located nearest our initial estimate as the final baseline axis.

may touch the baseline instead of the bottom of the bars. We therefore build a histogram of the top and bottom y-values of the candidate bars. The mode of this histogram represents the horizontal line that most bars touch, and we use the mode as an initial estimate of the baseline axis (Figure 3.5c). In practice we have found that noise, antialiasing, and other pixel level artifacts in the charts can sometimes lead to small inaccuracies (e.g., a few pixels) in our estimate of the position of the baseline axis. However, many charts draw the baseline x-axis as a solid line. Thus, similar to Yokokura and Watanabe [119], we further refine our baseline axis estimate by computing the y-gradient of the original chart and summing the rows of this image to form the y-gradient histogram (Figure 3.5d). We then treat the peak of this histogram located nearest our estimated baseline axis position as the final baseline

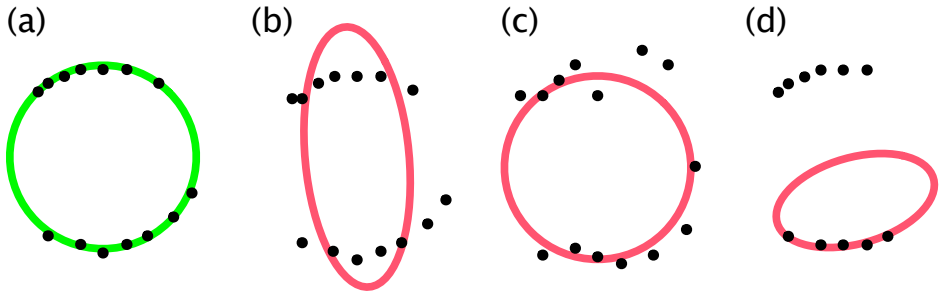


Figure 3.6: A low scoring ellipse and three high scoring ellipses. (a) A low scoring ellipse maximizes circularity, goodness of fit and coverage. (b) An ellipse with high circularity score, (c) high fit score, and (d) high coverage score.

position.

We do not extract the y-axis for bar charts. As we will show in the data extraction section, we can extract data values for the bars without knowing the location of the y-axis.

Recovering small bars

When we initially find rectangular shapes, we discard very small rectangles that have widths or heights less than 2 pixels. However, we have found that in a few cases these rectangles may be legitimate bars representing small values. Since bars in a vertical bar chart almost always have the same width, we recover small bars by first computing the average width of the remaining candidate bars and then re-introducing any bars we previously discarded with the same width (within 3 pixels). Finally, we discard any bars that are more than 8 pixels away from the baseline axis, and any bars whose width is 3 pixels more or less than the average width of the candidate bars. This step leaves us with the final set of candidate bars (Figure 3.3d).

3.3.3 Extracting Marks from Pie Charts

Mark extraction in pie charts involves two phases. We first fit an ellipse to the pie. We then unroll the pie and locate strong differences in color to locate the pie sector edges.

Fitting the pie

Although most pie charts are circular, some are elliptical. For greatest generality, we model the pie as an ellipse. Our goal is to fit the ellipse to the set of pixels at the edge of the pie. We start by thresholding the gradient magnitude of the chart to extract gradient pixels (i.e., pixels where the color changes sharply), as these are likely to lie at edges. We set the threshold such that gradient pixels comprise at least 1/30 of the image pixels, as we

empirically found this parameter to be sufficient to fit the ellipse. We use the text region tags to remove gradient pixels due to text.

While the set of gradient pixels is likely to include the pixels at the edge of the pie, it is also likely to include many additional pixels where the color changes sharply. Thus, we adapt the RANSAC [34] algorithm to robustly fit an ellipse to the gradient pixels that are most likely to lie at the edge of the pie while rejecting the other outlier pixels. Our algorithm works as follows: we first randomly select four gradient pixels and compute an ellipse that passes through them using Fitzgibbon et al.'s [35] direct least-squares ellipse fitting method. We then find the set of gradient pixels that are at most 10 pixels away from the ellipse and call them *inliers*. Next we check how well the ellipse explains the inliers by computing an ellipse score s, which is a weighted sum of three components: the circularity of the ellipse, how tightly it fits the inliers (average distance of an inlier to the ellipse), and coverage (how much of the ellipse is not near inliers). Lower scores indicate better ellipses. More formally, given an ellipse with major axis a, and minor axis b, we define the score as follows:

$$s = 2 * s_{\text{circularity}} + 0.5 * s_{\text{fit}} + 3 * s_{\text{coverage}}$$

We compute the circularity score by $s_{\text{circularity}} = 1 - \frac{b}{a}$. We compute s_{fit} by computing the average distance of the inliers to the ellipse. However, we prefer larger ellipses to smaller ellipses, since many pie charts have an outline which we want our fitted ellipse to contain. We therefore impose a penalty to inliers that lie outside the ellipse using the following equation:

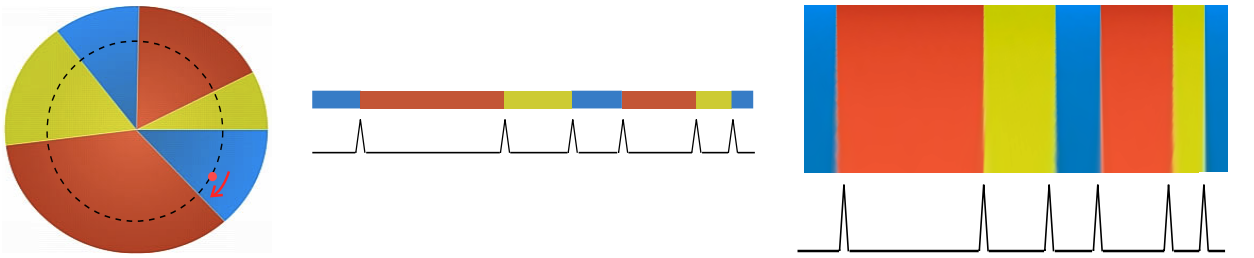
$$s_{\text{fit}} = d(\mathbf{x}) * \begin{cases} 1.2 & \text{if } \mathbf{x} \text{ is outside ellipse} \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

where $d(\mathbf{x})$ is the distance of inlier $\mathbf{x}$ to the ellipse. Finally, we compute s_{coverage} by computing the percentage of the ellipse that is covered by inliers. We estimate this value by projecting all inliers to the ellipse and counting the number of unique projected pixels on the ellipse. We then divide this number by the circumference of the ellipse and subtract from 1. Figure 3.6 shows examples of high scoring ellipses.

On each iteration, we keep the ellipse if its score is lower than the previous lowest score. We iterate this process 20,000 times, which we experimentally found to work well for the chart images in our corpora.

Locating sector edges

To extract the pie sector edges we first "unroll the pie": we sample an ellipse located inside the pie at 1000 evenly spaced angular intervals to create a one-dimensional ellipse image (Figure 3.7a). We then take the horizontal derivative of this ellipse image to identify strong changes in color. Color changes indicate transitions between pie sectors, and so the peaks in the derivative give us estimates for the sector edge locations (Figure 3.7b). To increase robustness of our estimates, we unroll multiple concentric ellipses and sum their horizontal derivatives (Figure 3.7c). Specifically, we unroll 396 ellipses whose axes evenly vary from 0.2 to 0.99 times the axes of the fitted ellipse (equivalent to spacing the scaling



(a) Chart (b) Unrolled ellipse and difference image (c) Unrolled pie using ellipses at different radii

Figure 3.7: Unrolling the pie. Consider the inner ellipse marked in (a). We unroll the ellipse by sampling 1000 points at constant angular intervals (b). Peaks in the horizontal derivative occur at sector edges. To improve edge estimation, we unroll multiple ellipses and sum their horizontal derivatives (c). Peaks in the summed horizontal derivatives occur at sector edges.

factors evenly by 0.01). We identify peaks in the summed derivatives by looking for zero crossings of its first derivative, which we find by convolving the summed derivatives with a derivative of a 3-pixel wide Gaussian. This smoothing handles noise and pixel-level artifacts. Some pies include thin borders between pie slices and the smoothing aggregates the peaks at either edge of the border into a single peak. Finally we retain all peaks that are more than one standard deviation above the mean of the summed derivatives.

3.3.4 Extracting Data from Bar Charts

In the data extraction step, our goal is to recover the data encoded by each mark. We assume that a mark encodes a tuple of data, where one dimension is quantitative and one is nominal. We recover these tuples by using the geometry of the extracted marks and the text region tags from the classification stage. The output of our data extraction step is a table that contains an ID and a data tuple for each mark.

To recover the data from a bar chart, we first infer the mapping between image space and data space. We assume a linear mapping, but recovering other mappings (e.g., logarithmic) should be straightforward. The linear mapping is fully defined by (1) a scaling factor between image space and data space, and (2) the minimum value (usually the value at the x-axis).

We first recover the scaling factor by considering the y-axis labels. We identify the y-axis labels that encode data values by finding text regions that are equidistant from the leftmost bar and line up vertically. We then estimate the scaling factor using each pair of value labels, as illustrated in Figure 3.8. We assume the labels were recovered by our text extraction procedure in the classification stage. The pixel distance between labels "5" and "10" is $d = 60$ pixels, and the estimated scaling factor is $60/(10-5) = 12$ pixels/data unit. We compute the scaling factor for each pair of labels and take the median as our final estimate. For this chart, the median scaling factor across all label pairs is 12.5 pixels/data unit.

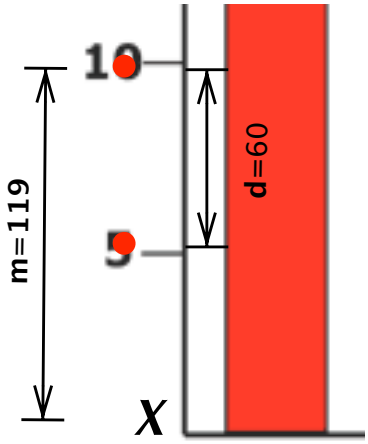


Figure 3.8: Computing the pixel/data unit scale for a vertical bar chart using the y-axis labels.

We then find the minimum value. We begin with the y-axis label vertically closest to the x-axis. If this label is "0", we set the minimum value to 0. Otherwise, we compute the minimum value using a similar procedure to computing the scaling factors. For each label, we estimate a minimum value by using the y-distance (in pixels) from the label's center to the x-axis, and the chart scaling factor. For example, using the location of the "10" label in Figure 3.8 we find the pixel distance is $m = 119$. Assuming a chart scaling factor of 12.5, we find $X = 119/12.5 - 10 = -0.48$. The actual minimum is 0; our result is not exact, but close. We set the chart's minimum value to the median of these minimum value estimates. For this chart, the median minimum value was -0.2.

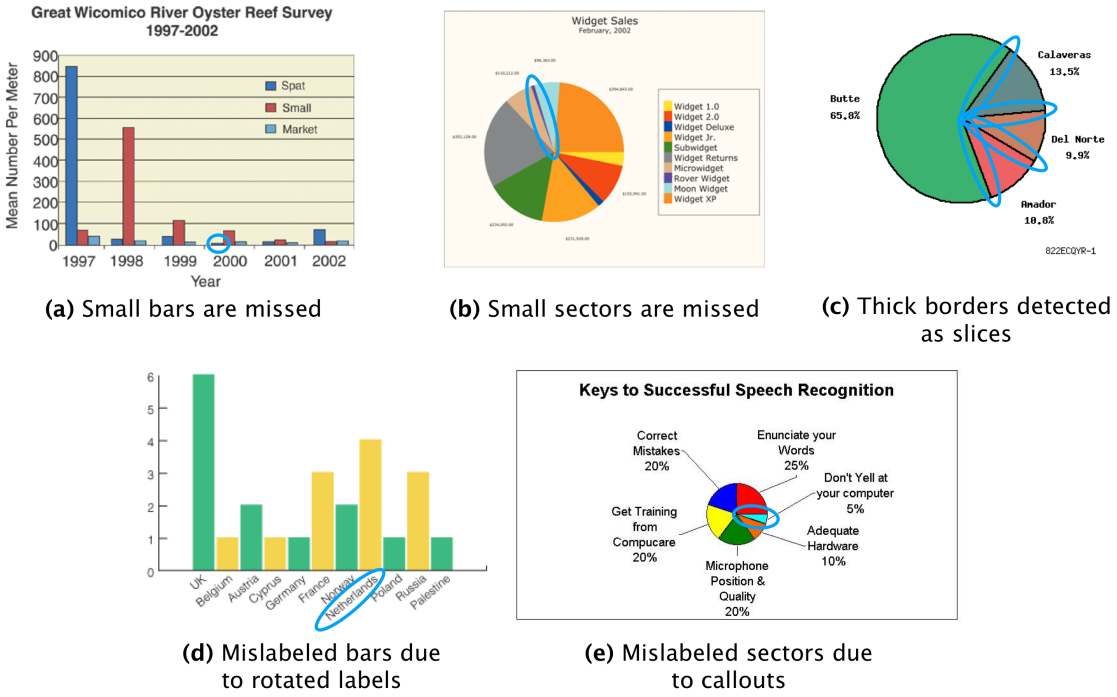
Finally, we assign a nominal value to each bar by associating it with the nearest label below the baseline x-axis.

3.3.5 Extracting Data from Pie Charts

Each sector of a pie encodes two data values: (1) a quantitative value representing the percentage of the whole pie covered by the sector, and (2) a nominal value representing the label of the sector. We compute the percentage as the angular extent between the edges of the sector. We treat the text label nearest the elliptical arc spanning the sector as the nominal label for the sector.

3.4 Results

To test our extraction algorithms, we used the subset of Prasad et al.'s [80] corpus that met our assumptions, which resulted in 52 bar charts and 53 pie charts. For this test we also assumed the chart type was known a priori, noting that our classification stage provides this information.



(a) Small bars are missed

(b) Small sectors are missed

(c) Thick borders detected as slices

(d) Mislabeled bars due to rotated labels

(e) Mislabeled sectors due to callouts

Figure 3.9: Mark and data extraction failures. If marks are very small, our algorithms fail to extract them. For example, we fail to extract the circled bar in (a) and the circled sector in (b). In pie charts with thick borders, we sometimes detect the borders as slices (c). Data extraction failures occur when marks are mislabeled, e.g., (d) when labels are rotated, or (e) if the chart places labels far from their associated marks, such as the circled marks.

Many of the charts in this corpus are small and of low quality, as its purpose was to test classification methods: the average chart dimension was 411 pixels. We hypothesized that our algorithms would be more effective on higher quality charts, so we gathered a separate set of high-resolution charts drawn from a web search, comprising 29 pie charts (with an average dimension of 780 pixels) and 20 bar charts.

To generate ground truth, we manually identified the marks and generated data tuples for each mark based on nearby text labels. We used sector labels (for pie charts) and axis labels under the baseline (for bar charts) as the nominal values. If a quantitative label was located near a mark we treated it as the quantitative value of the mark (e.g., Figure 3.11 top-left and Figure 3.12 top-left). Otherwise, we did not generate a ground truth quantitative value. For pie charts, we converted quantitative values to percentages. For bar charts, we directly used the labeled values.

Using the ground truth data, we found that ReVision successfully extracted all the marks for 41/52 (79%) of bar charts and 33/53 (62%) of pie charts. Most mark extraction failures occurred because we failed to detect small marks (Figure 3.9a, b). Our algorithms are designed to treat small, thin regions as decorative elements rather than marks that encode data. With relatively small chart images (on average, 342×452 pixels in our corpus) our algorithms can have trouble separating legitimate marks from these decorative elements. For pie charts with thick borders, we encountered the opposite problem – our algorithms would identify the borders as pie slices since when the pie is unrolled (Figure 3.9c). In these cases, our algorithm does not apply enough smoothing to remove the borders.

Accurate data extraction depends on accurate mark extraction. Focusing on the charts for which we were able to extract all marks, we accurately extracted data tuples for 29/41 (71%) of the bar charts and 21/33 (64%) of the pie charts. The most common error was incorrect association of a nominal label to a mark. Our simple closest-label heuristic for associating text labels with marks is especially prone to errors when labels are rotated (Figure 3.9d), or when marks are small and labeled with callouts (Figure 3.9e).

We could only generate quantitative ground truth for 12 of the bar charts and 17 of the pie charts for which we extracted the data. The remaining 17 bar charts and 4 pie charts did not have quantitative labels. Examining this subset of charts, we found that our extracted quantitative values were on average within 7.7% of the original data for bar charts and within 4.6% of the original data for pie charts. We believe that these are reasonable error rates given the small sizes of many of the charts, because we cannot achieve subpixel accuracy. Figures 3.11 and 3.12 show examples of successfully processed chart images and bar charts of the extracted data.

Running times varied widely with chart size. On a 2.4Ghz MacBook Pro with 2Gb of RAM, extracting marks and data from an average sized pie chart took 1,225s. Extracting marks and data from an average sized bar chart took 100s.

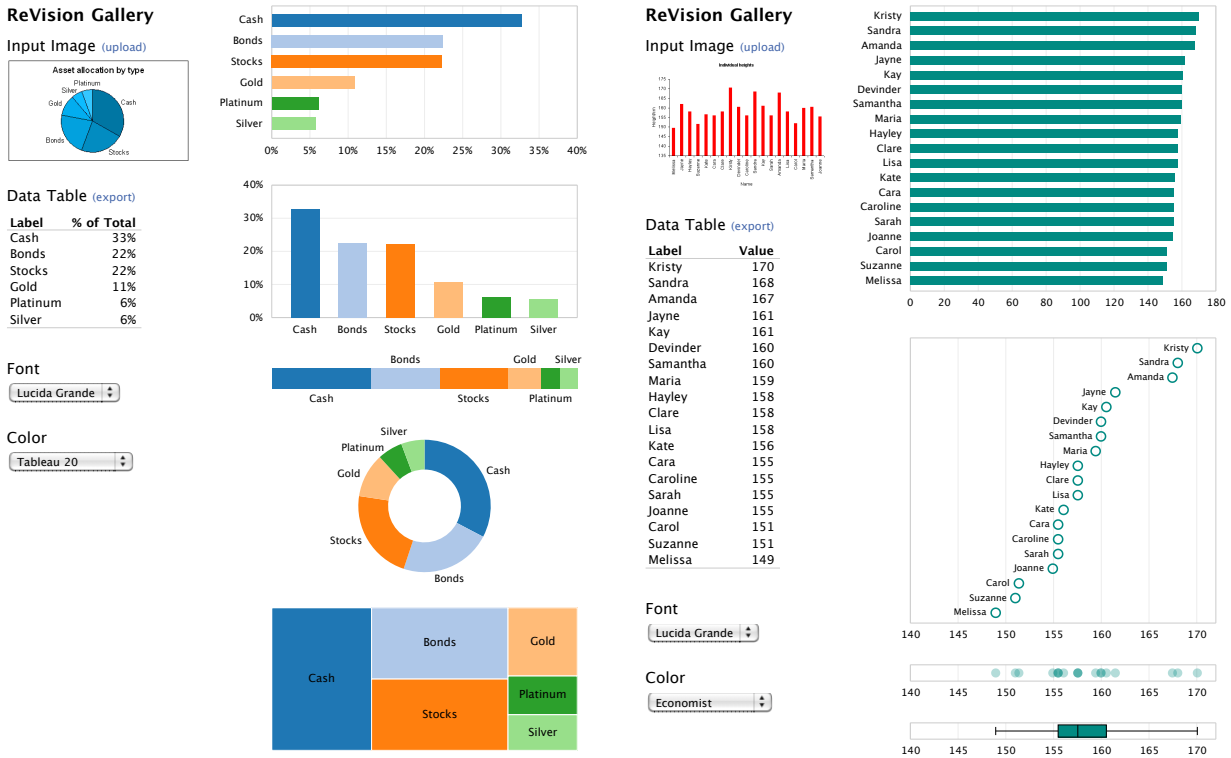


Figure 3.10: ReVision Design Galleries. Given an extracted data table, the gallery presents a variety of chart types, sorted by proposed perceptual effectiveness rankings [71]. Users can also select and compare color schemes and typefaces.

3.5 Stage 3: Redesign

The output of the data extraction process is a relational data table. ReVision uses this data to populate a gallery of alternative visualizations (Figure 3.10). We rank visual encodings by *effectiveness* [71] and display a variety of visualizations in a sorted gallery. ReVision chooses different visualizations depending on the input chart type and extracted data. For the input pie chart in Figure 3.10a, the gallery presents bar charts to support part-to-part comparisons and divided bar, donut, and treemap charts to support part-to-whole judgments [87]. For the input bar chart in Figure 3.10b, ReVision generates a bar chart and labeled dot plot to support comparison of individual values, and small dot and box plots to enable assessment of the overall distribution. Note that the y-axis of the input chart does not start at zero; ReVision's bar chart correctly incorporates a zero baseline and displays the data range with more appropriate charts (dot and box plots).

In addition, users can select and compare choices of font and color palette. ReVision includes palettes designed for well-spaced, perceptually discriminable colors [43; 98], as well as palettes from popular charting tools and news magazines. We generate charts using Protovis [11]; viewers can export the Protovis definitions for subsequent modification or

reuse. Alternatively, users can export the data to create their own charts using tools such as Microsoft Excel or Tableau [72].

3.5.1 Conclusion

We have presented ReVision, a system that classifies charts, extracts their graphical marks and underlying data table, and then applies perceptually-based design principles to automatically redesign charts. In particular, we described robust mark and data extraction algorithms for bar and pie charts, given certain assumptions.

Automated redesign is only one of many possible applications for a system that extracts data from charts. In the following chapters, we leverage ReVision's mark and data extraction algorithms to improve the usability of charts in other ways, such as by supporting cognitive and perceptual processes viewers undergo when reading a visualization (Chapter 4), and as a key component in our crowdsourcing pipeline to extract the references between charts and related text (Chapter 5).

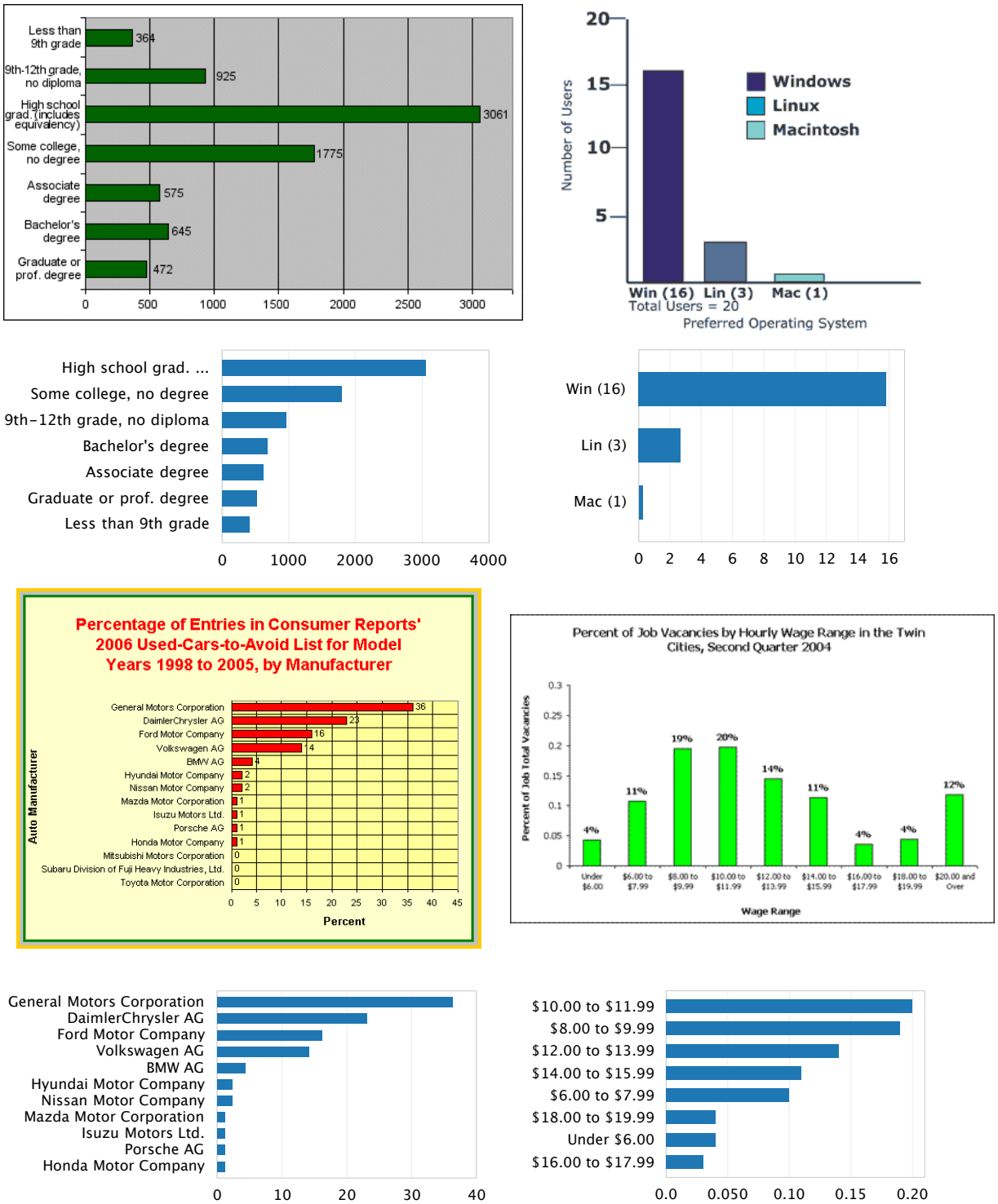


Figure 3.11: Example ReVision redesigns for input bar charts.

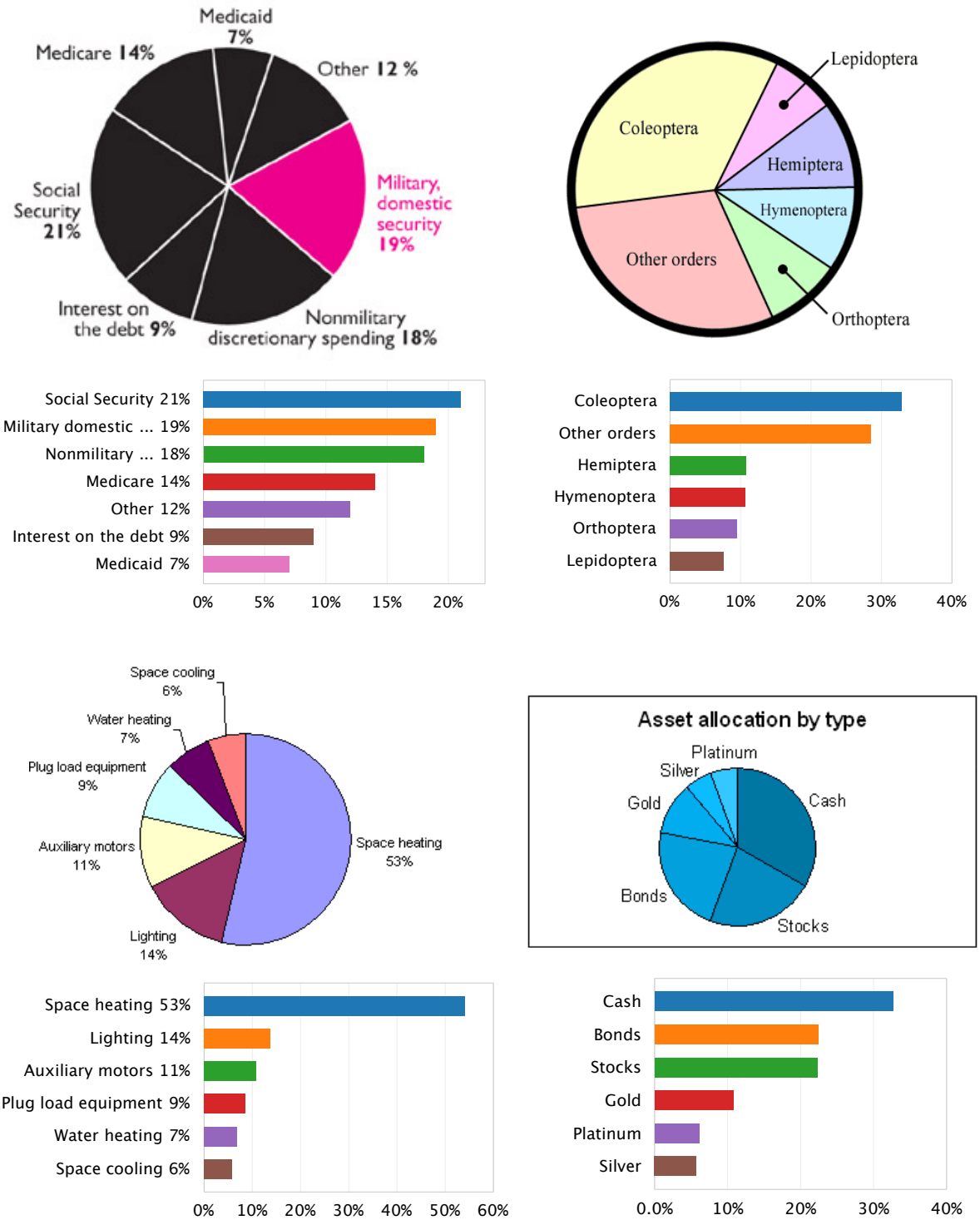


Figure 3.12: Example ReVision redesigns for input pie charts.

Chapter 4

Graphical Overlays

Reading a visualization involves a combination of perceptual and cognitive processes. Consider the task of extracting the value of the bar encoding the 2005 budget in Figure 4.1-1. The viewer must find the relevant bar, mentally project the top of the bar to a point on the y-axis, consider the nearest labeled tick marks and interpolate these numerical labels to compute the value at the projected point. While other types of charts (e.g., pie charts, line charts, scatterplots, etc.) use different visual encodings for the data, common chart reading tasks such as extracting, comparing and aggregating values, usually involve a similar sequence of perceptual and cognitive processes [61; 69; 78; 87]. However, many visualizations published in newspapers, reports, books and on the Web only support a subset of these processes.

In this chapter, we introduce *graphical overlays*—visual elements that are layered onto a chart to facilitate a larger set of perceptual and cognitive processes involved in chart reading. We identify five main types of overlays, each designed to support different processes. (1) *Reference structure* overlays, such as gridlines (Figure 4.1-2), aid the viewer in extracting and comparing values. (2) *Highlight* overlays draw the viewer's attention to certain marks by creating perceptual groups of marks. (3) *Redundant encoding* overlays allow viewers to extract numerical data in multiple ways and can be used to better depict some aspects of the data such as trends (Figure 4.1-3). (4) *Summary statistics* overlays depict aggregate information about the data set such as its mean, median or maximum (Figure 4.1-4). (5) *Annotation* overlays help viewers communicate and collaboratively analyze charts. While all five overlay types add visual elements to aid chart reading, they can also increase visual clutter. As a result, many published visualizations include few if any such overlays.

Additionally, we present an automated system that applies user-chosen graphical overlays to existing chart bitmaps. By targeting existing charts, our system allow users to tailor published visualizations to better support the chart reading tasks they wish to complete. Our system takes a bitmap chart, an overlay type, and an optional set of user-specified parameters as inputs and outputs a graphical overlay for the chart. Our approach is based on the key insight that generating most graphical overlays requires only knowing the properties of the visual marks and axes that encode the data and does not require access to the underlying data

Previously published by the author and Maneesh Agrawala in [59].

Figure 4.1: In this chart of the European Union's budget by the BBC [13] the original design (1) forces viewers to mentally project a line to the y-axis to extract values. (2) A gridline overlay provides visual anchors, which can simplify the process of extracting values. (3) A line overlay encodes the data redundantly but better illustrates the trends in the data across time. (4) Finally, a statistical summary overlay depicts the mean value of the data so that viewers can easily compare each year's budget to the average budget across the years. All of these overlays were generated by our system without access to the underlying data, based on automatic extraction of the chart's mark and axis properties.

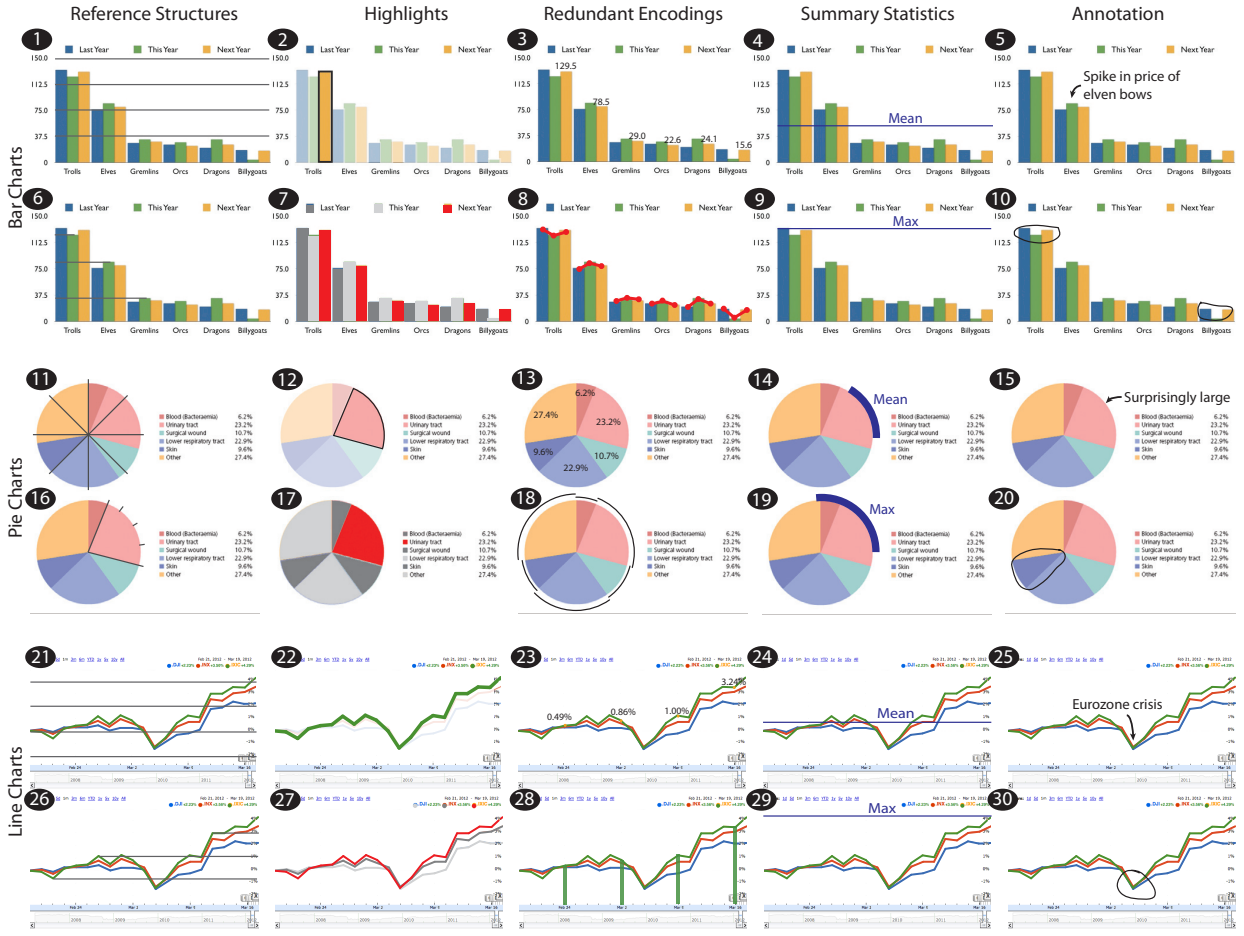


Figure 4.2: Examples of visual overlays organized by type. These overlays were manually generated to illustrate the different overlay types. See Figure 4.8 for results generated by our system.

values. Thus, our system analyzes the chart bitmap to extract only the properties necessary to generate the desired overlay. We support bar, pie, and line charts in our implementation. For bar and pie charts, we use the mark and data extraction algorithms from Chapter 3. For line charts, we use a manual extraction tool (Datathief [111]). We also show how our system can be used to generate interactive overlays and how it can place overlays underneath the marks in a chart.

4.1 Taxonomy of Overlays

Graphical overlays are designed to support the perceptual and cognitive processes used in chart reading tasks, such as extracting, comparing, or aggregating numerical values. We have analyzed a variety of charts published in books [109; 110] and online to identify five common types of overlays that each support a subset of these processes: reference structures,

highlights, redundant encodings, summary statistic, and annotations. Figure 4.2 contains two examples of each overlay type for bar, pie, and line charts. We first consider how these overlays support cognitive and perceptual processes when they are added to a visualization in static form, and then discuss how adding interaction to these overlays can further facilitate chart reading tasks (Section 4.1.6).

Although we focus on overlays that are layered on top of the marks in the base visualization, most overlay layers could alternatively appear below the marks as *underlays*. While overlays emphasize the content of the overlay, underlays emphasize the marks of the base visualization. Underlays support layering and separation better than overlays as viewers can more easily choose to ignore the underlay layer or direct their attention to it as necessary [99; 110] (Figures 4.7-5 and 4.7-6).

4.1.1 Reference structures

Overlays can provide reference structures [4] that are designed to help viewers recover the mapping between the visual encoding and the data, as shown in the first column of Figure 4.2. Reference structures such as gridlines directly facilitate the elementary perceptual processes of anchoring and projection [87], as shown in Figure 2.1, and thereby help viewers extract and compare numerical data values. Gridlines can be placed at regular intervals along an axis as in Figures 4.2-1 and 4.2-21, or emanate from specific marks to provide a more direct reference for the mark value as in Figures 4.2-6 and 4.2-26. Figures 4.2-11 and 4.2-16 show polar gridlines for pie charts. The former example is a direct analog of the regular gridlines in the context of pie charts, while the latter example regularly divides a specific pie slice at 5 degree increments to facilitate reading the percentage value of the slice relative to the whole pie.

4.1.2 Highlights

Overlays can highlight important marks, as shown in the second column of Figure 4.2. Such highlights draw the viewer's attention to specific marks by forcing them to share one or more distinguishing visual attributes (e.g., hue, saturation, texture, border color, drop shadows, etc.). The highlighted marks form a perceptual group based on the Gestalt principle of similarity [114]. Highlights can also act as deictic references for specific marks and thereby aid communication; viewers can refer to all the "red" highlighted marks rather than describing each one independently [48]. The overlays in Figures 4.2-2, 4.2-12, and 4.2-22 highlight certain marks by desaturating the other marks so that they appear closer to the color of the background. Figures 4.2-2 and 4.2-12 add a high contrast black border to a highlighted mark while Figure 4.2-22 thickens the highlighted line to further emphasize the chosen mark. The overlays in Figures 4.2-7, 4.2-17, and 4.2-27 re-color specific marks to a pre-specified highlight hue while setting the color of the other marks to different shades of gray.

4.1.3 Redundant encodings

Overlays can provide alternative encodings of the data, as shown in the third column of Figure 4.2. Such redundant encodings can help viewers extract values or they can emphasize trends in the data. Figures 4.2-3, 4.2-13, and 4.2-23 show overlays containing numerical data labels. Viewers can directly read the data value from the label instead of activating additional cognitive processes [78] to infer the encoded mark value using axis labels. Figure 4.2-8 shows an overlay that joins the tops of a group of bars together using lines. Figure 4.2-28 shows the converse for a line chart, where bars are drawn underneath data points. Lines communicate trends better than bars, whereas bars communicate individual data values more effectively than lines [78; 120]. Although redundant encodings show the same data, each encoding may be best suited to a different chart reading task. Finally, Figure 4.2-18 shows an overlay that contains arcs outside pie slices that emphasize the length of the outer edge of the slice. Viewers may perceive such length encodings more accurately than the angle or area encoding provided by the original pie slices [25].

4.1.4 Summary statistics

Overlays can contain visual elements that illustrate summary statistics of the numerical data, such as the mean, median, standard deviation, or global or local maxima or minima, as shown in the fourth column of Figure 4.2. Such overlays allow viewers to visually compare individual marks to a statistic based on the complete distribution of data values. These overlays save the user from performing time-consuming cognitive functions to mentally aggregate the data and compute the summary statistics [78; 108]. Overlays presenting the mean and max values of the data set are shown in Figures 4.2-4, 4.2-14, and 4.2-24 and Figures 4.2-9, 4.2-19, and 4.2-29 respectively.

4.1.5 Annotations

Overlays can contain annotations that provide contextual information or comments, as shown in the fifth column of Figure 4.2. These overlays can aid users in communicating about charts or collaboratively analyzing charts as they allowing viewers to create arrows and other deictic references to marks [48; 58]. Figure 4.2-5, 4.2-15, and 4.2-30 show overlays that contain text annotations associated with marks. Figure 4.2-10, 4.2-20, and 4.2-30 show overlays that contain freehand annotations.

4.1.6 Adding interaction to overlays

Interaction in graphical overlays can help reduce visual clutter and increase the variety of tasks an overlay supports by allowing users to interactively specify which marks an overlay should target. For example, consider the overlay in Figure 4.2-6, where horizontal lines are drawn from bars to the vertical axis to support projection [87] (Figure 2.1-c). If lines were drawn from every bar, the display would quickly become cluttered and the viewer would

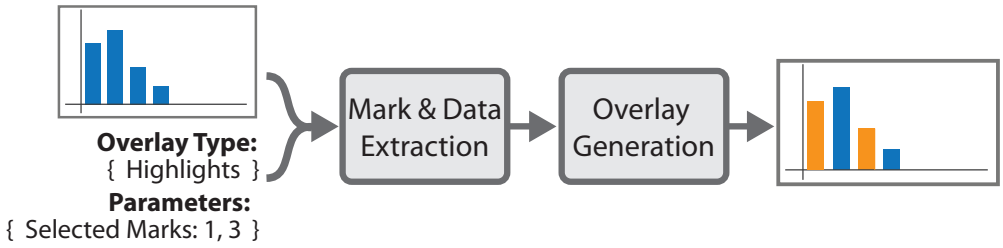


Figure 4.3: Overview of the system. Our system takes a chart bitmap, overlay type, and author-specified design parameters (e.g., grid spacing, highlight hue, selected marks to highlight, etc.) as inputs. It then extracts the necessary marks and data and passes this information to the overlay generation component, which outputs a graphical overlay on the input bitmap.

need to perform additional cognitive processes to separate the lines from each other [69]. An interactive version of this overlay could allow the viewer to specify which of these lines to draw by clicking on bars, thus reducing visual clutter while still supporting the viewer's specific task.

Interaction can also provide direct manipulation of overlay elements, which would support the superimposition and anchoring processes [87] (Figures 2.1-a and 2.1-d) and decrease the amount of spatial cognition the viewer must apply [108]. For example, the polar gridlines in Figures 4.2-11 and 4.2-16 could move with the viewer's cursor and thereby allow the viewer to align the gridlines with a specific slice.

4.2 A System for Producing Visual Overlays

We have developed an automated system that applies a user-chosen graphical overlay to existing charts. We refer to the person creating the overlay as the overlay author. Our implementation supports all of the overlays shown in Figure 4.2. Figure 4.3 shows an overview of our system. The input to our system is a chart bitmap and the specific overlay type. While our system provides default values for each of the overlay design parameters (e.g. gridline spacing, highlight hue, line thickness, and font style, etc.), authors can optionally set these parameters through a web-based user interface (Figure 4.4). Our system then extracts the necessary marks and data from the chart and outputs an overlay over the existing chart. We first discuss the properties required to construct each graphical overlay and the tools we use to obtain those properties. We then describe our implementation of the graphical overlays.

4.2.1 Chart Properties for Overlay Generation

One approach to building graphical overlays is to generate them directly from the underlying data values. However, correctly recovering the data values from a chart bitmap is difficult; for example, ReVision achieves only 47.9% accuracy in extracting data, where the

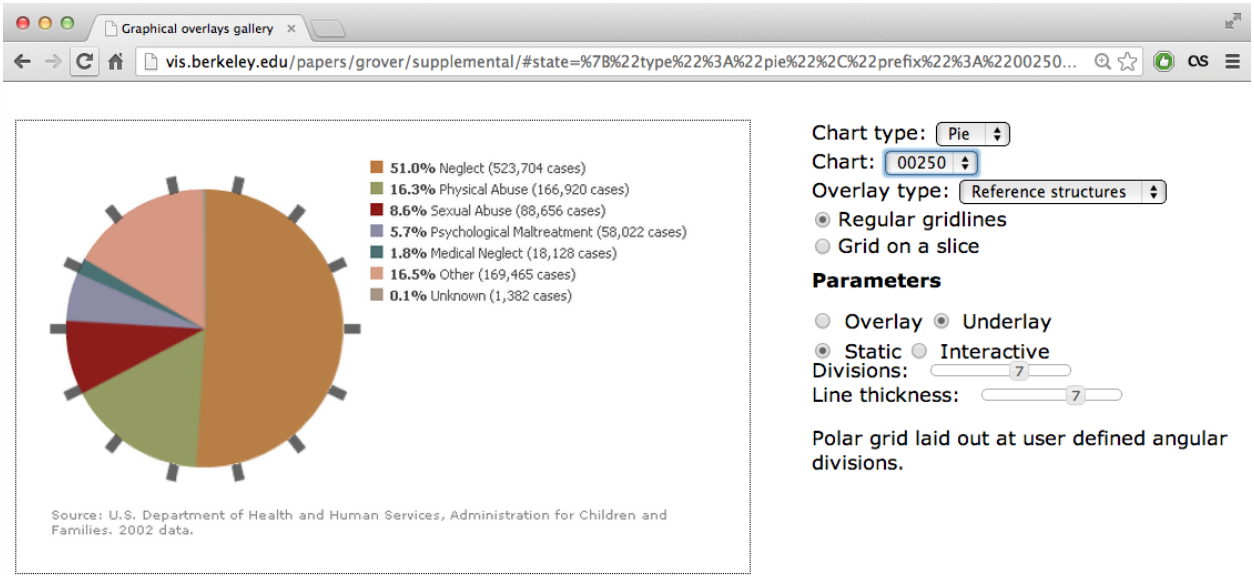


Figure 4.4: User interface for authors to interactively modify graphical overlay design parameters such as line thickness, font size, the number of divisions in a gridline overlay, highlight hue, etc.

extracted values are within 6% of the true data values on average (Chapter 3.4). Recovering the properties of marks and axes is much easier; ReVision achieves 71.4% accuracy in extracting these elements.

Our system takes advantage of the insight that most graphical overlays only require access to the mark and axis properties of the base visualization, and can be generated without access to the underlying data. For each type of overlay we consider whether it requires mark properties, axis properties or access to the underlying data.

Mark properties

Mark properties encompass the retinal variables used to visually encode the data [8]. Our system primarily uses two mark properties; (1) the location of the mark boundary and (2) the color of the mark. Knowledge of mark boundaries allows our system to generate pie chart gridlines (Figures 4.2-11 and 4.2-16). Similarly, the summary statistics overlays only require access to mark boundaries. For example, the position of a mean line in a vertical bar chart (Figure 4.2-4) can be computed by averaging the highest boundary position of each bar. Highlighting overlays (Figure 4.2 second column) require both the boundary and color of marks. In some cases mark properties can directly represent the data that encodes the mark; Figure 4.2-13 shows an overlay containing percentage value labels for each slice. The mark boundary provides direct access to the length of the outer slice boundary and the percentage of the pie covered by the slice.

Axis properties

Axis properties encompass the locations and orientations (i.e., the endpoints) of the axes in bar and line charts. The axis of a pie chart is implicitly based on a polar grid centered on the pie and is therefore extracted as part of the mark boundary property. We assume bar and line charts are two-dimensional and have no more than two axes, one vertical and one horizontal, explicitly depicted in the chart. This assumption implies that the chart area is given by the extents of the two axes. Regular gridlines (Figures 4.2-1 and 4.2-21) are an example of overlays that only require axis properties — we draw horizontal or vertical lines that span the chart area at regular intervals. Other overlays require both axis and mark properties (e.g., Figures 4.2-4, 4.2-6, 4.2-9, 4.2-24, 4.2-26, and 4.2-29).

Data

The data encompasses the numerical data values that the chart marks encode. Although any of the graphical overlays can be generated if the underlying data is known, the only overlays in Figure 4.2 that require access to the data are 4.2-3 and 4.2-23, which contain numerical data labels for a subset of the marks.

The first part of our system involves extracting the marks or data required for the author-chosen overlay (Figure 4.3). We use a mix of automatic and semi-automatic extraction techniques. For bar and pie charts, we use ReVision [83] to extract the relevant chart properties. ReVision applies computer vision and machine learning techniques to identify the chart type, extract the graphical marks, and infer the underlying data. For line charts we use Datathief [111] to semi-automatically extract chart properties. Our overlay system is designed to operate independently from the technique used to extract chart properties and could be combined with other extraction techniques [2; 10; 118]. Alternatively the author could provide the necessary properties directly to the system as input.

4.2.2 Overlay Implementation

The main component of our system is overlay generation (Figure 4.3), which we implemented in HTML5 and JavaScript. We render graphical overlays by placing a `<canvas>` element over the chart bitmap. We describe implementation details for each type of overlay, including the author-specified design parameters necessary to create each overlay. For brevity we describe our implementation of bar chart overlays assuming vertical bar charts, but our implementation includes the corresponding algorithms for horizontal bar charts.

Reference structures

Regular gridlines (Figures 4.2-1 and 4.2-21) require the author to provide a gridline direction (horizontal or vertical) and the number of divisions. Our system computes the chart area bounding box from the extracted axis properties and draws equally spaced lines in the

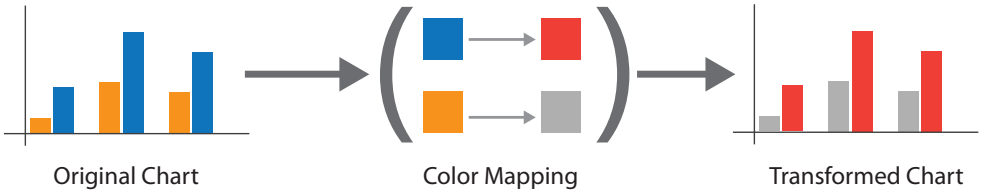


Figure 4.5: One way our system highlights marks is by performing a color transformation. We apply a color mapping function to the chart; highlighted marks are colored red, while other marks are colored shades of gray.

author-specified direction to divide the chart into the author-specified number of divisions. For example, Figure 4.2-1 is generated using horizontal gridlines and four divisions. For a pie chart (Figure 4.2-11) the gridlines are arranged radially and the author only has to specify the number of divisions. Some overlays draw gridlines that emanate from or target individual marks (Figures 4.2-6, 4.2-16, and 4.2-26). In these cases the author must specify the target marks by clicking on them. Our system uses hit testing based on the mark boundaries to identify the clicked marks.

Highlights

All overlays containing highlights require authors to first select the marks or data points they wish to highlight. Our system generates two variations of overlays containing highlights. Figures 4.2-2, 4.2-12, and 4.2-22 highlight marks by desaturating non-highlighted marks. We use the mark boundaries to produce this effect. First, we trace the boundaries of all the non-highlighted marks and fill the boundaries with a semi-transparent white overlay. By only desaturating the areas of the chart within the boundaries of non-highlighted marks, we preserve the color of any background elements. We then draw a black line on the boundary of the highlighted marks for bar and pie charts. For line charts, we increase the width of the highlighted line.

The overlays shown in Figures 4.2-7, 4.2-17, and 4.2-27 highlight marks by performing a color transformation (Figure 4.5). Given a set of author-selected marks to highlight, our system finds all pixels in the chart bitmap that match the original colors of the target marks. It then modifies the color of these pixels to the author-specified highlight color (such as red, in these examples). Finally, it converts all the remaining pixels to their grayscale values by averaging their RGB components. This approach ensures that the highlighted marks are perceptually distinct from the non-highlighted marks. Moreover, because it transforms pixel colors across the entire chart, it ensures that color-based legends correctly match the graphical marks even after the transformation. However, this approach may also unintentionally re-color important chart elements such as as text or background graphics.

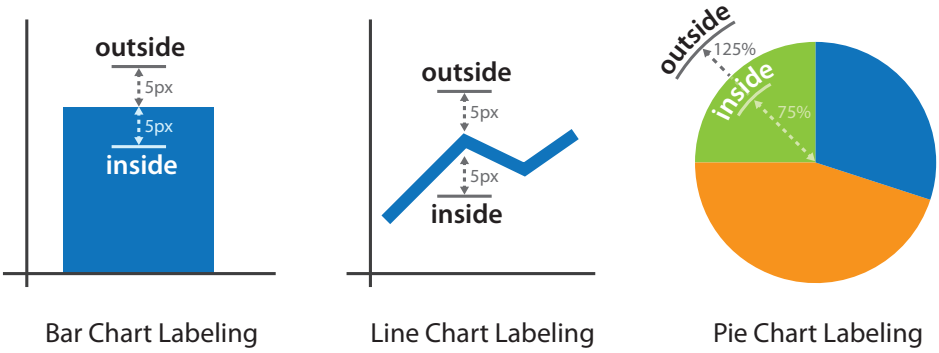


Figure 4.6: Our system allows authors to specify whether labels are placed inside or outside marks. For line charts, this choice translates to above or below a mark.

Redundant encodings

Overlay authors can choose to add numerical data labels or an alternative visual encoding for all of the marks in the chart (the default) or to an author-selected subset of marks. For data label overlays (Figures 4.2-3, 4.2-13, and 4.2-23) authors can optionally specify where each label should be placed relative to a mark boundary (Figure 4.6). For example, in a bar chart the author could specify that the label should be placed inside the bar, which translates to centering the label 5 pixels inside the top of the bar. Our system also provides default label layouts; for bar and line charts, it centers labels 5 pixels above each mark, while for pie charts it centers data labels radially within each pie slice and puts them at 75% of the radius from the center of the pie.

To generate alternative visual encodings (Figures 4.2-8, 4.2-18, and 4.2-28) our system relies on information about the mark boundaries. For the overlay shown in Figure 4.2-8, the author must select each group of marks to connect with a line. Our system then generates a circular point at the center of the top of each mark and draws lines connecting the circular points. For the overlay shown in Figure 4.2-18, our system uses the radius of the pie to draw circular arcs outside of the pie chart. For each slice our system draws a circular arc that spans the outer boundary of the slice and whose radius alternates between 3 and 5 pixels greater than the radius of the pie. Finally, for the overlay shown in Figure 4.2-28, our system draws rectangles whose fill color and height are the same as the author specified points on the line. The thickness of the bars is an author-specified parameter.

Summary statistics

Our system can create overlays depicting the mean, median, global maximum, global minimum of the data using just the mark and axis properties. For bar and line charts (Figures 4.2-4,4.2-9, 4.2-24, and 4.2-29), our system computes these statistics in image space using the heights in pixels of each mark, as given by the mark boundaries. The statistic is computed in pixel units and our system simply draws a line on the chart at the computed

height and places the statistic label just above the line, centered horizontally in the chart area. For pie charts (Figures 4.2-14 and 4.2-19), our system computes the statistics in angular extent using the mark boundaries of each slice. It then draws a thick circular arc outside the pie which subtends the angular extent of the computed statistic and places the statistic label just outside the center of the arc. The thickness of the summary statistic line (in a bar chart) or arc (in a pie chart) is an author-specified parameter.

Annotations

Our system implements two types of annotation layers: text annotations that are automatically associated with marks (Figures 4.2-5, 4.2-15, and 4.2-25) and freehand annotations (Figures 4.2-10, 4.2-20, and 4.2-30). For text annotations, our system allows authors to click on the chart and enter text. It then computes the nearest mark to the bounding box of the text and draws an arrow from the bounding box to the mark. An alternate interaction, although our system does not currently implement this, is to allow the user to manually specify the direction of the arrow. For freehand annotations, our system gives authors a black, pen-like tool to create arbitrary annotations.

Interaction and Layering

Interactive overlays allow end-users to directly manipulate and move the information displayed in the overlay. Our system supports such viewer interaction for most of the overlays shown in Figure 4.2. For example, the interactive version of the polar gridline overlay Figure 4.2-11 rotates based on the viewer's mouse cursor position relative to the center of the pie (Figure 4.7-1). Our system also allows viewers to interactively select target marks for the highlighting overlays (Figures 4.2, second column) and the redundant encoding overlays (Figures 4.2, third column). Similarly, viewers can select one or more marks (as in Figures 4.2-6, 4.2-16, and 4.2-26) and our system will generate gridlines emanating from just the selected marks. Our system also implements snapping to marks in interactive overlays. For example, holding the shift key in an interactive overlay containing highlights will highlight the mark nearest to the mouse cursor (Figure 4.7-4).

Finally our system can layer the information designed for an overlay *under* the marks to produce an underlay. Our system first renders the graphical overlay (as we have described above) and then draws the sections of the image bitmap that lie within each mark boundary on top of the overlay layer (Figure 4.7-6). While this approach works well for elements such as gridlines that extend beyond the boundaries of the marks, it may not produce the desired effect when the elements lie within a mark, as for some highlight overlays. However, we believe that alternative underlay templates could be designed to handle such cases.

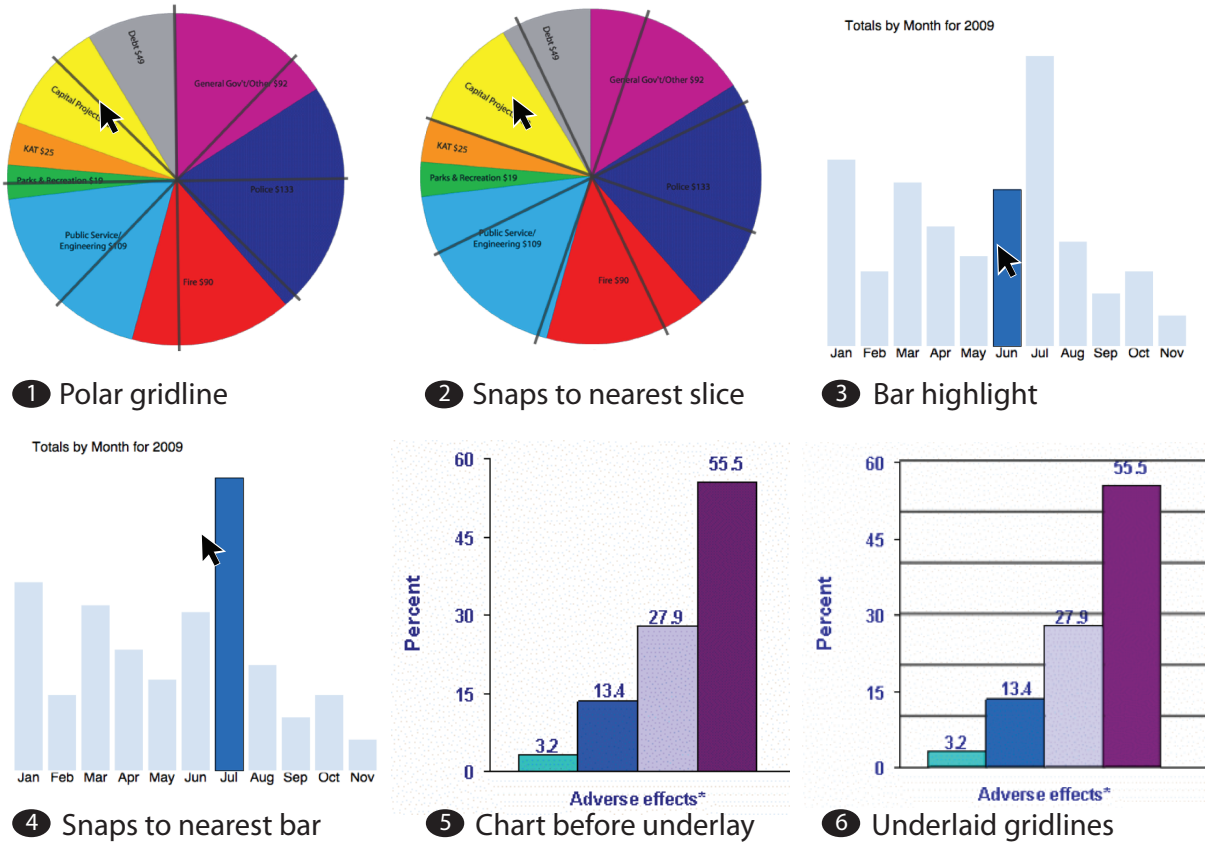


Figure 4.7: Interactive overlays (1-4) and underlays (5-6) generated by our system. (1) The user can reposition a polar gridline by moving the cursor. (2) When the user holds the shift key, the gridline snaps to the slice nearest to the cursor. (3) The user can highlight a mark by mousing over it. (4) When the user holds the shift key, the overlay highlights the nearest mark to the cursor. (5) A bar chart without any overlays. (6) Our system creates an underlay effect by re-rendering the areas of the chart that lie within mark boundaries after rendering the gridlines.

4.3 Results

We have used our system to generate each type of overlay shown in Figure 4.2 for 32 bar, 53 pie, and 7 line charts drawn from the Web. Figure 4.8 presents a subset of these results for 15 different charts (original chart is shown on top and overlaid chart is shown on the bottom). We include a larger set of both static and interactive overlays generated by our system on the web at `http://vis.berkeley.edu/papers/grover/supplemental`.

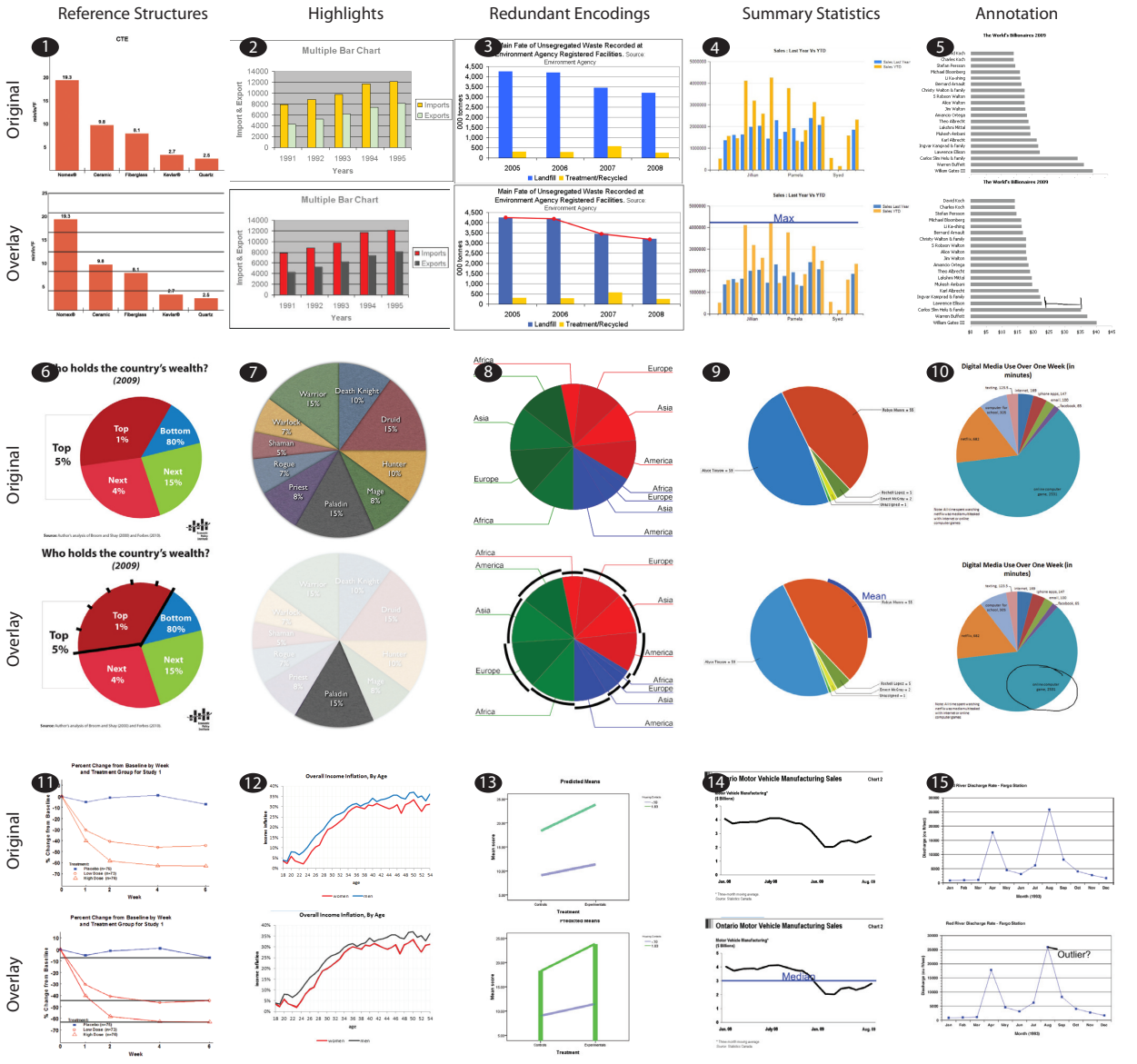


Figure 4.8: Example static overlays generated by our system for 15 different charts drawn from the Web. The original chart is shown on top and the overlay is shown on the bottom. For each case we apply one of the overlay types from Figure 4.2. We apply only one overlay to each chart. Some of the original charts contain gridlines or numerical data labels (e.g., #2 and #3) and we have applied a different overlay to them.

4.3.1 Reference structures

Figures 4.8-1, 4.8-6, and 4.8-11 show example reference structures. The original chart in Figure 4.8-1 did not contain gridlines, so we overlay a regular gridline with five divisions to

aid projection and extraction of the data values. In Figure 4.8-6, we overlay tics at regular 22.5 degree intervals for a selected pie slice to facilitate anchoring and extraction of the slice angle. In Figure 4.8-11, we overlay horizontal gridlines from a few selected data points of interest to also aid projection.

4.3.2 Highlights

Figures 4.8-2, 4.8-7, and 4.8-12 show examples of highlights. In Figures 4.8-2 and 4.8-12 we use the color transformation approach to facilitate visual grouping and draw the viewer's attention to the red highlighted elements. While this approach converts most pixels outside the selected marks to shades of gray, it properly preserves the mapping between the legend and the color of both the highlighted and non-highlighted marks. In Figure 4.8-7 we apply a highlight overlay that desaturates non-highlighted marks. This approach preserves the hue and texture of the pie slices.

4.3.3 Redundant encodings

Figures 4.8-3, 4.8-8, and 4.8-13 show examples of redundant encodings. In Figure 4.8-3, the overlaid red line connecting the blue bars illustrates the trend in the data over years better than the bars alone. In Figure 4.8-8, the overlaid arcs facilitate extracting the value encoded by the pie slice based on arc length rather than slice angle or area. The arcs are especially useful for the very small slices. In Figure 4.8-13, overlaid bars emphasize the values of selected points along the x-axis, which is especially useful here because the x-axis encodes a categorical variable.

4.3.4 Summary statistics

Figures 4.8-4, 4.8-9, and 4.8-14 show overlays that contain visual representations of the max, mean, and median respectively. In each of these cases, the statistic was computed using all of the marks. However, the overlay author can choose to compute these statistics using a subset of the marks, for example a single line in a line chart.

4.3.5 Annotations

Figures 4.8-5 and 4.8-10 show freehand annotations. In Figure 4.8-5, the annotation author has marked the large jump between the three lowest bars in the chart and the other bars. In Figure 4.8-10, the author has drawn attention to the text in one of the slices. Figure 4.8-15 shows a text annotation that for the sharp peak in the line chart.

Our system also implements interactive versions of many of these overlays. Figure 4.7-1 depicts an interactive gridline overlay for a pie chart that is positioned based on the viewer's cursor position. Figure 4.7-3 depicts an overlay that highlights the mark underneath the cursor by desaturating the other marks. Our interactive overlays also implement snapping.

When the viewer holds the shift key, the gridline snaps to the nearest pie slice (Figure 4.7-2), or the system highlights the nearest bar to the cursor (Figure 4.7-4).

4.4 Limitations

While our overlay generation system can produce high-quality overlays for many existing charts, it also has a few limitations. Our system does not consider visual elements that fall outside of the marks and axes. Such elements include legends, axis labels, background graphics, background colors, etc. In some cases our overlays may interfere with these elements. In Figure 4.9-1, for example, the original chart bitmap contains two textboxes describing the underlying data. Our highlight overlay desaturates the parts of both textboxes that lie within the non-selected pie slices, making it difficult to read parts of the text.

Although our system does not require the underlying data to generate most overlays, it does require accurate knowledge of mark and axis properties. Inaccuracies in the location of mark boundaries can produce visual artifacts in some overlays. Figure 4.9-2 shows a highlight overlay that contains artifacts at the bar boundaries due to slight inaccuracies in boundary extraction from the ReVision [83] system. While overlay authors could manually fix such inaccuracies today, we also expect that automated mark extraction techniques will improve and produce pixel-accurate boundaries.

Some of our overlay designs have specific limitations. The color transformation overlay (Figures 4.8-7 and 4.8-12) assumes a constant fill color within a mark. Antialiasing or compression artifacts violate this assumption, and can causes artifacts at the edges of marks (Figure 4.9-3). Finally, we use a simple layout algorithm for placing data labels, which can result in label-label overlaps.

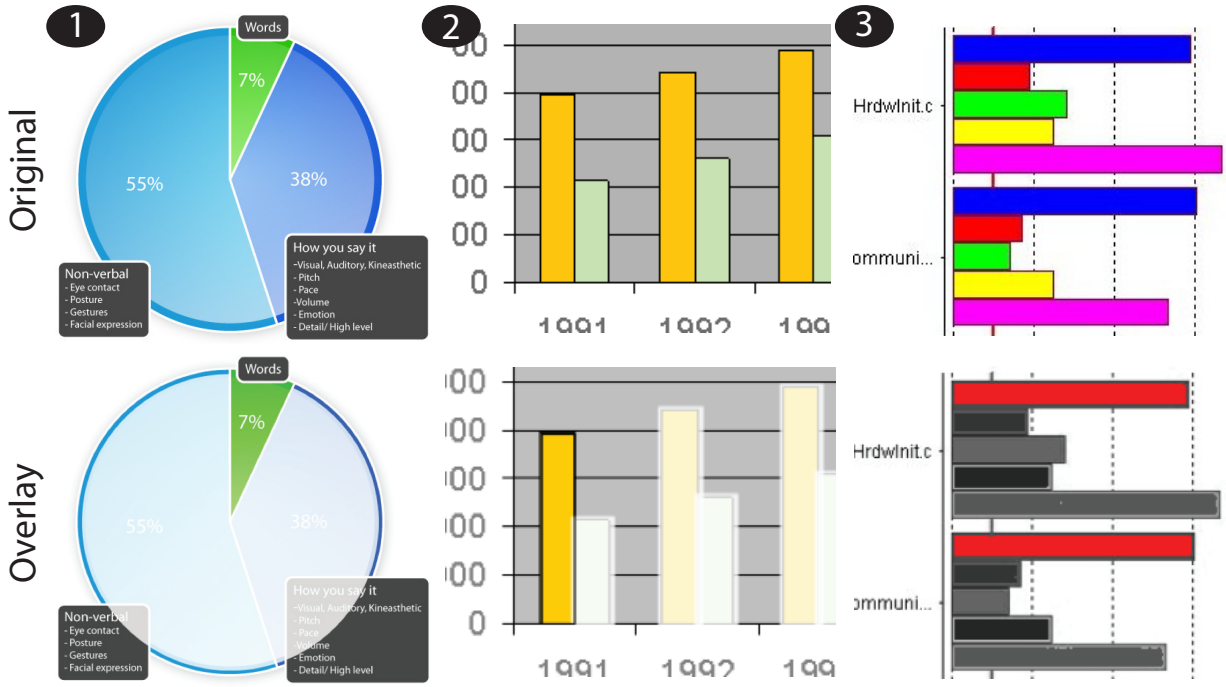


Figure 4.9: Limitations of our overlay system. (1) Desaturation affects the text boxes. (2) Mark boundary errors cause desaturation to extend beyond the true mark boundary. (3) Color transformation does not capture all the pixels inside marks due to antialiasing or compression noise. Note that (2-3) are cropped versions of the original overlays.

Chapter 5

Extracting References Between Text and Charts

Up to this point, we have considered techniques to extract information from and augment charts using only the chart alone. However, charts abound in blog posts, news articles, and academic papers, where they may emphasize key points presented in the text. Charts can also provide additional data, not mentioned in the text, to give readers more context and allow them to make their own inferences. Thus, for readers to fully understand such a document they must parse all of the *references* between the text and the corresponding visual marks (e.g., bars, lines, points, pie slices, etc.) in the charts.

Yet, identifying such references between the text and the chart can be challenging. Consider the example in Figure 5.1 from a Pew Research report [77]. The text explains that *"Half or more in 13 of 21 nations surveyed believe that most people can succeed if they are willing to work hard."* To find the corresponding nations in the chart, the reader must identify the countries for which the orange bar (most succeed if work hard) is longer than the blue bar (hard work no guarantee). Often, as in this case, the text only refers to a subset of the data in the chart and the reader must perform complex visual comparisons to identify the correct subset. In other cases, the text may paraphrase values in the chart and require the reader to bring external information to bear. For example, the text may use the term "EU" to refer to a subset of the European countries in the chart.

In this chapter, we present a crowdsourcing pipeline that takes a document containing text and one or more charts as input and extracts the references between the text and the chart. We then show how we can use the extracted references in a document reading application that allows users to highlight phrases in the text and see the corresponding marks in the chart.

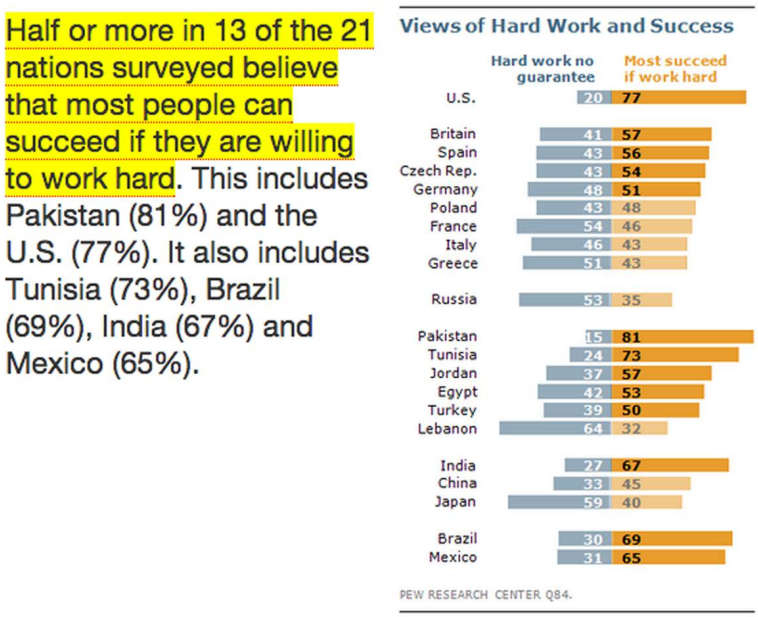


Figure 5.1: Example of a reference between the text and a chart from a Pew Research report [77]. The highlighted text (yellow background) refers to the 13 bar segments highlighted in the chart (saturated orange). We use our crowdsourcing pipeline to generate these references and display them here in our interactive document viewing application. The application lets users interactively select the text and it automatically highlights the corresponding marks in the chart.

5.1 Reference Extraction Pipeline

In a document containing text and charts, there are two kinds of references: (1) text phrases may refer to data and (2) visual marks may refer to data (Figure 5.2). The referent in both cases is one or more tuples (or rows) of an underlying data table. To solve the reference extraction problem we must recover both of these types of references for the input document. In this work we focus primarily on the first goal and rely on a combination of prior methods [83] and manual techniques to achieve the second goal. Throughout the paper, we will use the term *reference* to mean either the correspondence between a set of phrases (in the text) and data tuples or a set of visual marks (in the chart) and data tuples.

Our reference extraction pipeline (Figure 5.3) takes a document containing text and charts as input, and outputs the references between the text and charts. It consists of three main stages: a *pre-processing* stage in which we set up the crowdsourcing task, a *crowdsourcing* stage, in which we ask a group of workers to extract the references and another group of workers to vote on their correctness; and a *clustering and merging stage*, in which we combine the worker generated references into a unified reference set.

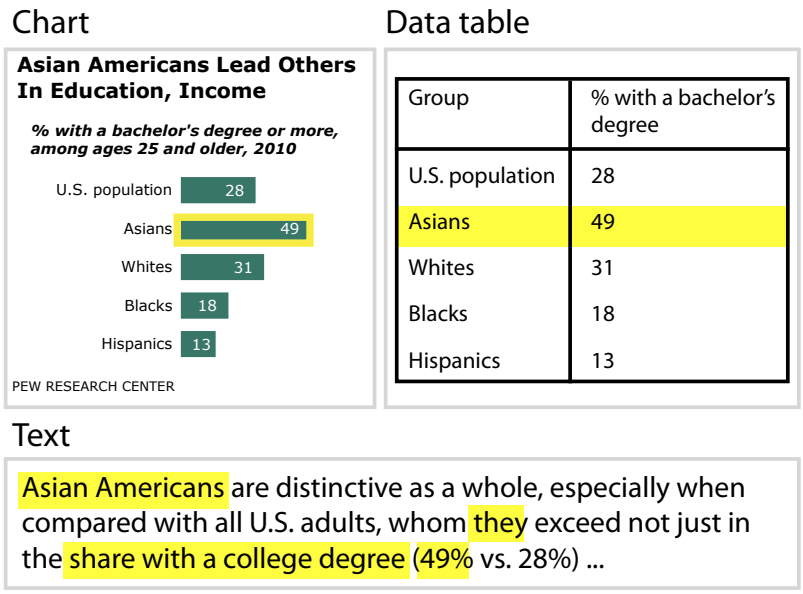


Figure 5.2: Text phrases (bottom) and visual marks in a chart (top-left) refer to data tuples (or rows) of an underlying data table (top-right). To extract the references we must identify the relationships between the text phrases, the visual marks and the data tuples.

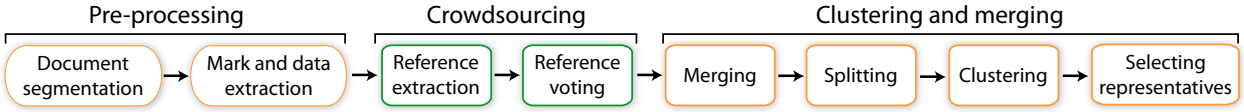


Figure 5.3: The three stages in our proposed pipeline for extracting references. Orange components are algorithmic or manual, while green components are crowdsourced. Given a document (left), we segment the paragraphs and charts and extract the marks and data in a preprocessing stage. We then ask workers to extract the references and vote on their correctness in the crowdsourcing stage. Finally, we cluster and merge the worker references into a unified set of references.

5.2 Stage 1: Pre-processing

In the pre-processing stage of our reference extraction pipeline we set up the crowdsourcing task by segmenting an input document into paragraph-chart pairs and extracting the marks and data table from the charts.

5.2.1 Segmenting the document into paragraph-chart pairs

One of the challenges in crowdsourcing is to design relatively small microtasks that workers can complete quickly in good faith [57]. Because reading a long, multi-paragraph document can be time consuming, we design the crowdsourcing task so that each crowd

worker only extracts references for a paragraph-chart pair (see next section on Stage 2 of our pipeline). Thus, in the pre-processing stage we split the input document into paragraphs and then manually pair each paragraph with the charts that are related to it (i.e., the text in the paragraph refers to the chart).

While we perform the pairing manually, this task is likely to be amenable to crowdsourcing. For example we could ask crowd workers to mark paragraphs and charts that are related. It may also be possible to automatically compute the pairing by analyzing the document layout as well as standard text references to figures (e.g., "see Figure 2"). However, in this work we focus on the key problem of extracting the references between text phrases and data tuples rather than on pairing the paragraphs with the charts.

5.2.2 Mark and data extraction

To extract references between the text surrounding a chart and the data tuples encoded in the chart, we must first recover the data table from the chart. We assume the charts in the input document are bitmaps, so we recover the data table in two steps. We first analyze the chart to identify the locations of all of the data-encoding marks (e.g., bars in a bar chart) as well as the chart axes. We then analyze the marks themselves in relation to the axes to recover the underlying data values.

We use the techniques from Chapter 3 to automatically extract the marks, axes and data values from simple bar charts. For more complex charts (e.g., stacked bar charts and grouped bar charts), we built a simple graphical interface for interactively annotating the marks and data values, and for associating the marks with the data values. We also use this interactive tool to correct errors or missing output from the automatically extracted marks and data. This mark and data extraction step is unnecessary if the author provides the information, or if a chart is dynamically generated via a visualization toolkit that exposes this information.

5.3 Stage 2: Crowdsourcing Reference Extraction

In the crowdsourcing stage of our reference extraction pipeline we ask workers to mark text phrases and the corresponding data-encoding marks for a set of paragraph-chart pairs. To maintain high-quality work we train the workers and regularly check the accuracy of their references on a small set of *gold tasks*. Finally, we ask another group of workers to vote on the correctness of the references produced by the first group of workers.

5.3.1 Reference extraction microtask

Figure 5.4 shows our microtask interface for extracting references between the text and data. The microtask includes a paragraph of text and a chart. We ask workers to select one or more text phrases (shown highlighted in yellow) and click on the corresponding set

of marks in the chart (shown highlighted with red outlines). Since the pre-processing stage gives us the mapping between marks and data tuples we can link the worker selected text phrases to the relevant data tuples and thereby form the complete reference between text and data.

Clicking the "Add Reference" button adds the reference to a list shown at the bottom of the task (Figure 5.4c). Workers can remove references from the list by clicking the **x** button and can add as many references as they wish before clicking the "Submit" button to finish the task. Thus, the worker can create one or more references for each paragraph-chart pair and each reference relates a set of words in the paragraph to a set of data tuples.

We experimented with an alternative microtask design in which we presented workers with the data table we recovered in the pre-processing stage and asked them to directly select the data tuples corresponding to their selected phrases. In pilot testing this interface we found that scrolling through a large data table to find the corresponding tuples was both tedious and error-prone. Selecting the relevant marks in a spatially compact chart was far easier for most workers and produced more accurate references.

5.3.2 Quality control

Crowdworkers do not always produce accurate, high-quality work. They may not understand the task, they may be lazy and do as little work as possible, or they may maliciously introduce errors [6]. We have designed our microtasks to control for work quality in two ways; (1) we require workers to pass a training task before they can submit references, and (2) we intersperse the regular tasks with *gold tasks* where the correct answer is known a priori [65; 76] so that we can check that a worker completed the task correctly. In this section we assume the existence of paragraph-chart pairs for we have a gold standard set of references as well as a quantitative distance measure for comparing worker generated references to the gold standard. We will describe how we created the gold standard references and the distance measure in the following section on Gold References and Reference Comparison.

Training workers

First-time workers must pass a training task before they can submit work for our reference extraction pipeline. The training task first describes the reference extraction task and presents a paragraph-chart pair with a corresponding set of gold references. It then gives the workers another paragraph-chart pair for which the gold standard references are known but this time asks them to extract the references themselves. We check for correctness by comparing the worker generated references to the gold standard references using our distance measure (see section on Reference Comparison). If the distance is high (> 0.5 out of a max of 1.0), we ask the worker to extract the references again and resubmit. Workers may submit references as many times as they wish but can only move on to the main task once they have completed the training task accurately.

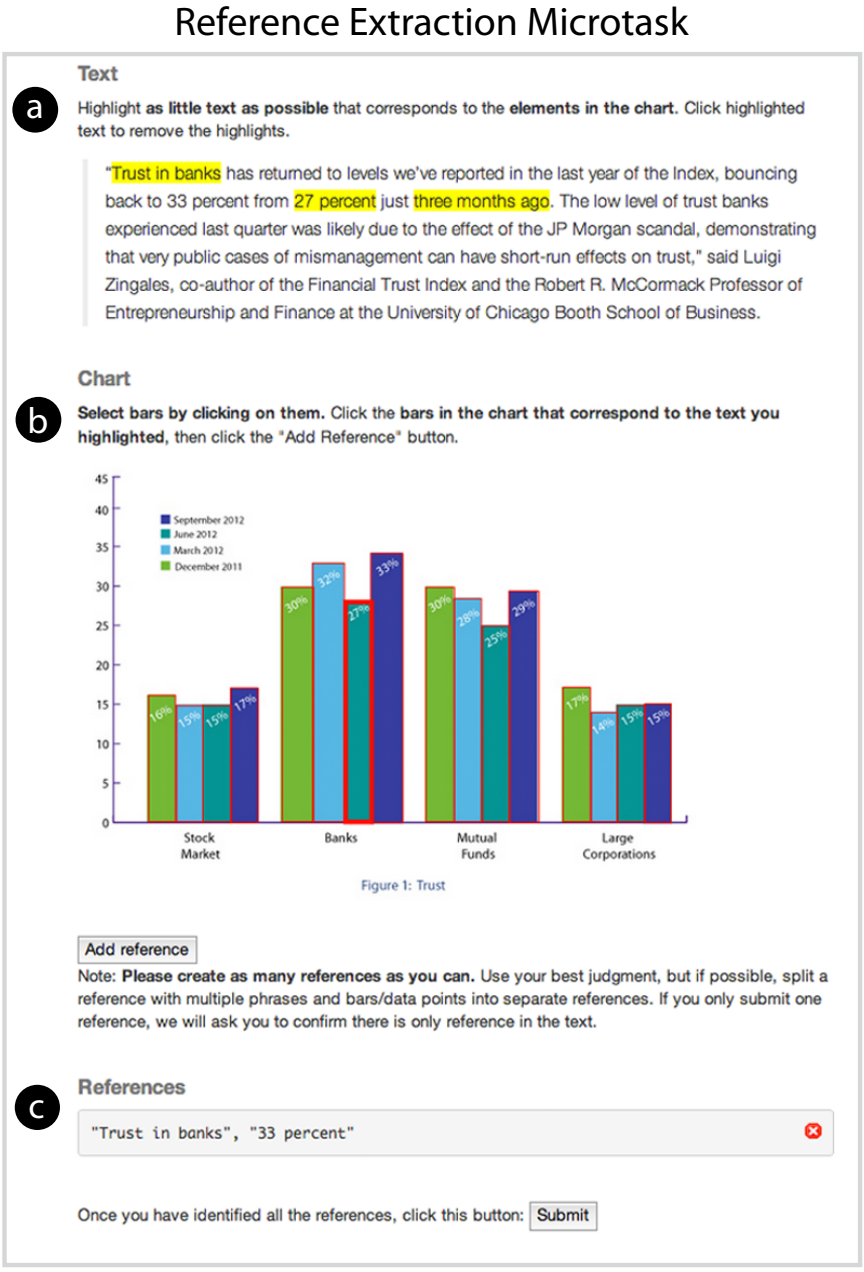


Figure 5.4: The reference extraction microtask presents a paragraph (a) and a chart (b). Workers must select text phrases (highlighted in yellow) and click on the corresponding visual marks in the chart (thick red outline). The thin red outlines in the chart indicate selectable marks. Clicking "Add reference" adds a row to the list at the bottom of the chart (c) showing the text of the reference. This list holds all of the references the worker has already created for this paragraph-chart pair. Workers can delete references they created earlier by clicking the x button.

Interspersed gold tasks

Workers may perform well on our training task but still perform poorly on other, more difficult tasks. They may also complete later tasks in bad faith. We therefore continue to monitor worker performance by asking them to complete two reference extraction tasks within each microtask: one for which we have gold standard references and one for which we do not. We randomize the ordering of the gold and non-gold tasks. If the worker correctly extracts the references for the gold standard task (i.e., the distance between the worker reference set and the gold reference set < 0.46), we keep their result for the non-gold; otherwise, we discard their result. We chose the 0.46 threshold empirically in conjunction with the clustering threshold (described later) and found it to give good results in practice. In the event that a worker has already completed all of the available gold tasks, we only show the worker the non-gold task, and we use their previous performance on the gold tasks (the average distance between their sets and the corresponding gold sets) to decide whether to keep or discard their result.

5.3.3 Reference correctness voting microtask

Finally, we ask another set of workers to vote on the correctness of some of the extracted references. If three or more workers submit identical references (i.e., the reference comparison distance between them is 0), we assume that the reference is correct since at least three people independently extracted it. In this case we do not ask other workers to vote on its correctness. For each remaining reference, we ask five additional workers to vote on whether the reference is correct.

Figure 5.5 shows the reference correctness voting microtask interface. For each reference, a worker sees a chart with highlighted marks and a paragraph of text with highlighted phrases. They then answer the question "Do **all of the highlighted bars** refer to **all of the highlighted text?**"

As with the extraction task, workers must pass a training task before voting on the validity of a reference. We present workers with a description of the task and examples of good and poor references. We then require them to correctly rate four references to pass training. As an additional quality-control measure, workers must also submit the number of highlighted bars, the first highlighted word, and a way to improve the reference for every reference voting microtask they complete. These tasks ensure that workers carefully read and consider each reference. We automatically discard the work if either the number or word are incorrect, and we discard the answers workers give for improving the references.

5.4 Gold References and Reference Comparison

To assess the quality of worker generated references, we created gold standard references for a set of paragraph-chart pairs. We also designed two distance measures for comparing

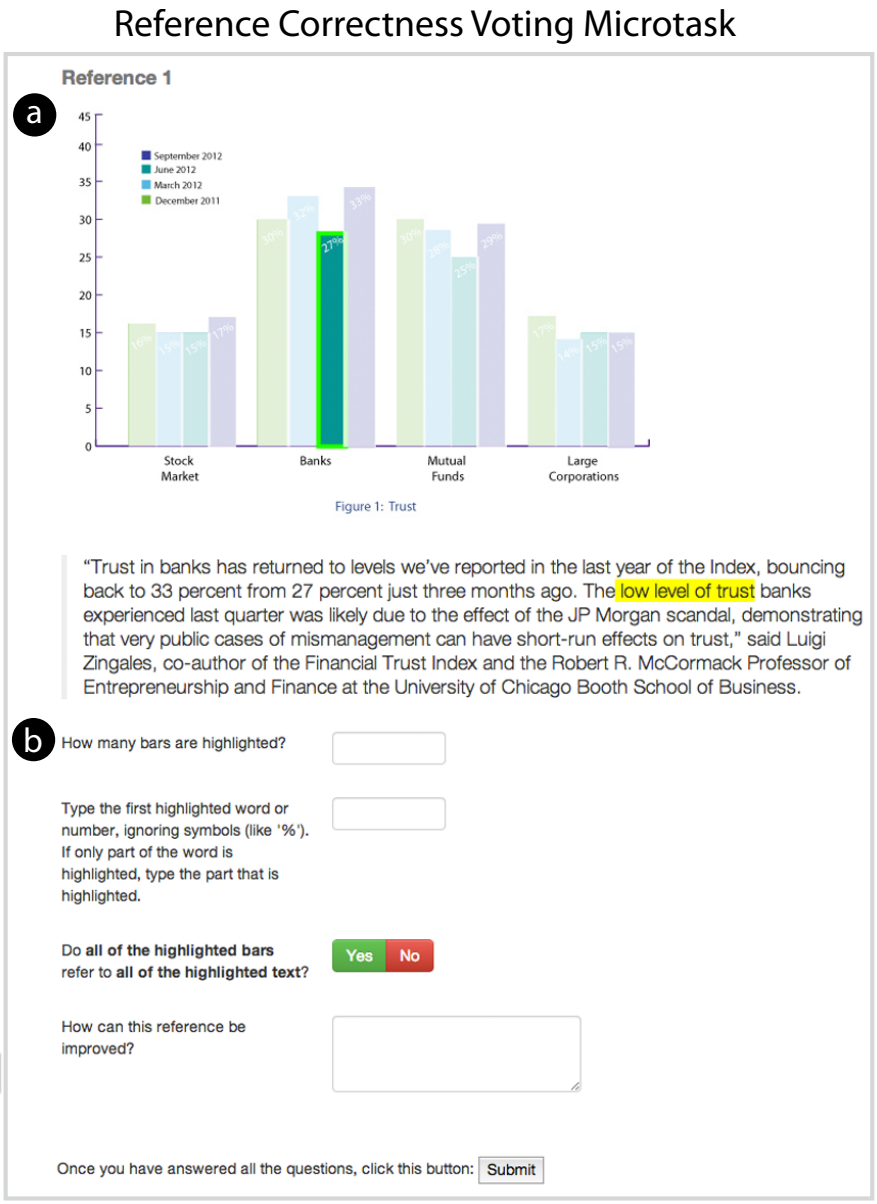


Figure 5.5: The reference correctness voting microtask presents workers with a reference by highlighting a set of marks in the chart (outlined in thick green with all other marks faded) and as well as a set of phrases in the paragraph (yellow background) (a). Workers must answer basic questions about the reference, whether all of the highlighted bars are related to all of the highlighted text, and how the reference can be improved (b). The basic questions and free response are designed to force workers to pay attention to the task.

worker generated references to the gold references: a distance between a *single* worker reference and a *single* gold reference, and a distance between a *set* of worker references and a *set*

of gold references.

5.4.1 Creating gold standard references

In any paragraph-chart pair only a subset of the text phrases refer to data. We designed our gold standard references to capture the minimal set of text phrases and data tuples that uniquely correspond to one another. An example of a such a *minimal reference* is shown in Figure 5.2; adding more text could only increase the number of tuples in the reference while removing phrases would make the correspondence with the data tuple ambiguous. Figure 5.1 illustrates another minimal reference in which the minimal text refers to multiple data tuples.

Two experts (the first author and a post-doc from our lab) created a gold standard set of references using an iterative process. We jointly drafted an initial set of guidelines for creating minimal references, independently extracted minimal references from five paragraph-chart pairs and then jointly revised the guidelines based on the results. Our two experts converged on the following guidelines for our gold references:

- Each reference should contain as little text as necessary to explicitly specify a relation to one or more tuples.
- Each reference should have a different phrase set.
- Ignore text that refers to the chart as a whole (e.g., the phrase "religious extremism" in a chart where every tuple is related to religious extremism).
- In the case of an ambiguous phrase that refers to an ambiguous subset of a chart (e.g., "the rich" or "the poor"), make a best effort guess.

We then separately extracted references for the remaining corpus of paragraph-chart pairs (see Section 5.6 for a description of our corpus). Finally, we came to a consensus on the paragraph-chart pairs where we disagreed on the content or number of references to produce the final set of gold references. Using the distance measure described in the following section to compute an average distance between the two expert sets, we found reasonable agreement ($\mu = 0.23, \sigma = 0.25$) – in fact, for 29 of the paragraph-chart pairs, we extracted identical phrases and tuples.

5.4.2 Distance measures for comparing references

For each paragraph-chart pair, each worker can submit one or more references linking text phrases to data tuples. To compare the worker generated references to the gold standard references we first compute the distance between each worker generated reference and each gold reference. We then combine these single reference distances to compute the overall distance between the set of worker generated references and the set of gold standard references. In the following discussion, we use lower case letters w and g to refer to a single worker or gold reference respectively. We use upper case letters W and G to refer to sets of references.

Distance: Single worker to single gold reference

A reference is composed of a collection of phrases (or equivalently, a collection of words), and a corresponding collection of data tuples. To compute the distance between two references we separately measure the similarity between the phrases and between the data tuples using the F_1 score [21] and combine the scores as follows

$$d(w, g) = 1 - \left(F_1^{text}(w, g) \cdot F_1^{data}(w, g)\right). \tag{5.1}$$

The F_1 score is a common statistical measure of the similarity between two collections [21]. It is based on measures of precision and recall and is computed as follows,

$$\text{precision}(w, g) = \frac{|w \cap g|}{|w|}, \tag{5.2}$$

$$\text{recall}(w, g) = \frac{|w \cap g|}{|g|}, \tag{5.3}$$

$$F_1(w, g) = \frac{2 \cdot \text{precision}(w, g) \cdot \text{recall}(w, g)}{\text{precision}(w, g) + \text{recall}(w, g)} \tag{5.4}$$

In these equations, we have overloaded the notation and use w and g to denote either the collection of text phrases or the collection of data tuples for the worker or gold reference respectively. To compute $F_1^{text}(w, g)$ we treat w and g as collections of text phrases and to compute $F_1^{data}(w, g)$ we treat them as collections of data tuples.

Our single reference distance measure $d(w, g)$ lies in the range $[0, 1]$, where 0 denotes an exact match, and 1 denotes no similarity in either the text phrases or data tuples of the worker and gold reference.

Distance: Set of worker to set of gold references

A worker's set of references W is close to a gold set of references G if each reference in the worker's set has a low distance to the nearest reference in the gold set and vice versa. Figure 5.6 illustrates how we compute the distance $d(W, G)$ between these sets of references for a paragraph-chart pair. For each worker reference we find the nearest gold reference; that is, for each $w \in W$ we find the $g \in G$ such that $d(w, g)$ is minimized. This gives us a correspondence between each worker reference and a gold reference. However, we may be left with some unmatched gold references. So, for each unmatched gold reference $g \in G$ we find the nearest worker reference $w \in W$ by minimizing $d(w, g)$. Finally, we compute $d(W, G)$ as the sum of the distances between corresponding worker and gold references, normalized by the total number of correspondences.

Handling references with extraneous text

As described earlier, we designed our gold standard references to be minimal. Although our reference extraction microtask asks workers to submit such minimal references, in practice

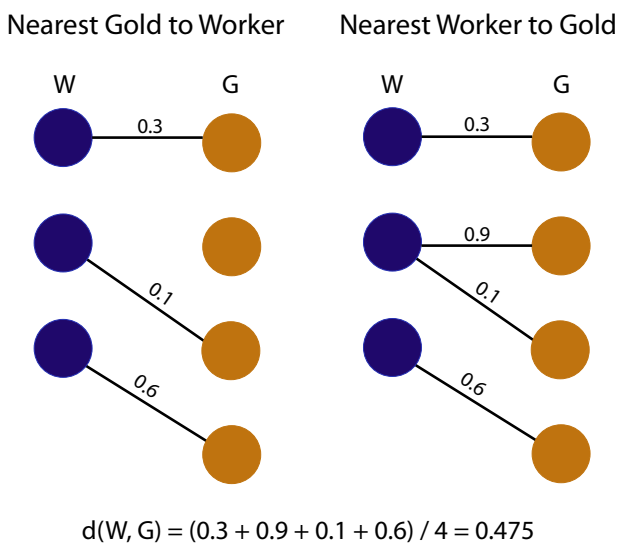


Figure 5.6: Computing $d(W, G)$ for a set of worker references W (blue) and a set of gold references G (orange). (left) For each worker reference, we find the nearest gold reference. Here, one gold reference remains unmatched. (right) So for each such unmatched gold reference, we find the nearest worker reference. Finally, we compute the distance by averaging the distances between corresponding worker and gold references.

we have found that workers often include *extraneous text phrases* that do not have any impact on the relationship between the text and the data tuples. For example, Figure 5.2 shows a paragraph-chart pair with a minimal set of text phrases corresponding with the tuple (Asians, 49). Suppose a worker marks the following phrases

> *Asian Americans are distinctive as a whole*, *especially ...* *share with a college degree* *(49% vs. 28%)*

as corresponding with this tuple. In this case the worker's phrases contain all of the text in the minimal reference, but also add the extraneous words "*are distinctive as a whole*". The worker's reference remains unambiguous even though it is not minimal.

Since such extraneous words are common in worker references and do not increase ambiguity we ignore extraneous words in our distance computations. Specifically, for each paragraph-chart pair we construct the set of extraneous words by starting with all the words in the paragraph and removing all of the text phrases that appear in the gold standard references. We then remove the extraneous words from each worker reference before computing a distance.

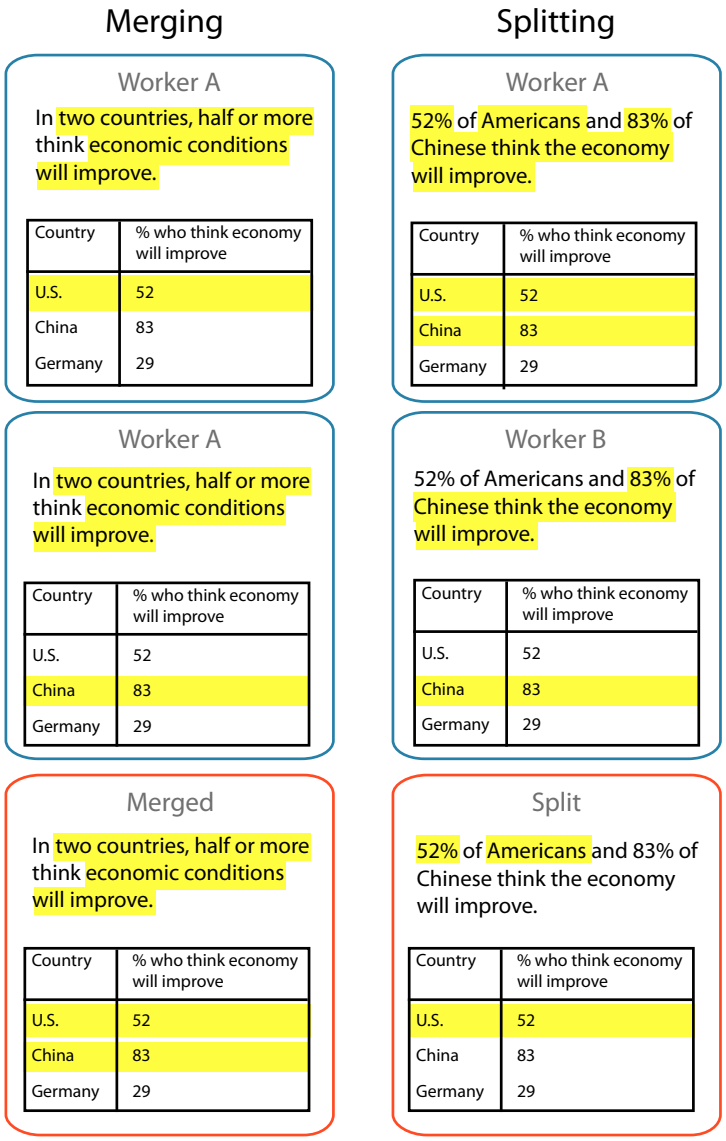


Figure 5.7: (left) For each worker, we merge references with the same text but different tuples by taking the union of the text and tuples. (right) We split larger references into smaller, more minimal references by finding references that are subsets of one another. Here, Worker B's reference (middle) is a subset of one submitted by Worker A (top). We subtract B's reference from A's to obtain a more minimal reference (bottom).

5.5 Stage 3: Clustering and Merging References

The set of references generated by a single worker for a given paragraph-chart pair can be inaccurate: they may miss references and form an incomplete set, contain incorrect ref-

erences, or include references that are not minimal. To reduce such problems we we ask multiple workers (10 in our experiments) in the crowdsourcing stage to independently generate a set of references for each paragraph-chart pair. In the clustering and merging stage we algorithmically combine the independently generated references to form a set of *unified references*. The goal of this stage is to integrate the most accurate parts of the worker generated references. We perform the following operations:

1. We merge references that contain the same set of text phrases but different data tuples into a single reference (Figure 5.7 (left)).
2. We compare references across all workers to identify references that are subsets of one another. We then split the larger references into smaller, more minimal references using the subset relationship. (Figure 5.7 (right)).
3. We cluster all of the resulting references based on their similarity.
4. We choose a single representative reference for each cluster resulting to form the output set of *unified references*.

5.5.1 Merging references containing the same text phrases

Workers sometimes create different references that include the same collection of text phrases but different data tuples. We assume each such reference is incomplete and merge them into a single reference by taking the union of their phrases and data tuples. In practice, we have found that we produce more accurate results if we merge references even when there are small differences in their text (i.e., a few extra or missing words). Therefore, we merge references that share 95% of their words. Figure 5.7 (left) shows an example of two worker references that we merge into a single reference using this process. The merged reference contains all of the words and tuples from both worker generated references.

5.5.2 Splitting references based on subsets

When comparing references across multiple workers it is common to find references that are subsets of one another. Consider the worker generated references shown Figure 5.7 (right). Let a_t and a_d denote the collections of text phrases and data tuples respectively for reference a (Figure 5.7 (right, top)) and let b_t and b_d denote the corresponding collections for reference b (Figure 5.7 (right, middle)). In this case b is a subset of a because $b_t \subseteq a_t$ and $b_d \subseteq a_d$. That is, the text and data tuples of b are contained in a.

We split the larger reference a into smaller, more minimal references by iteratively subtracting all such subset references b from it. That is, we replace a with $a - b$ by replacing a_t and a_d with $a_t - b_t$ and $a_d - b_d$ respectively. We continue subtracting all subset references b in this manner until either a_t or a_d is empty or we have subtracted all of the subset references b. The resulting reference a after subtraction is usually much closer to minimal (Figure 5.7

(right, bottom)). We repeat this subtraction process for every large reference a for which we find subset references b. Note that to properly account for the subtracted references in the fourth step of clustering and merging (choosing representative references) we duplicate each subtracted reference b and add it to the pool of worker generated references.

5.5.3 Clustering references by similarity

Next, we cluster the references. Since we ask multiple workers to submit references for each paragraph-chart pair, we expect that many references will be similar. To eliminate such redundancy we first cluster together the similar references and then in the next step we choose a representative reference for each cluster. This approach is designed to ensure the that final set of unified references are as distinct as possible.

To cluster the references we first form a graph in which each reference is a node and we create an edge between references a and b if $d(a, b) < 0.32$. This condition ensures that connected references are similar to one another. We chose the 0.32 distance threshold empirically and found it to give good results in practice. To build the clusters we compute maximal cliques for this graph using the following greedy algorithm. We initialize the first clique with the reference containing the most tuples (in case of a tie we pick randomly). We then iterate through the remaining references that have not been added to a clique, adding a reference to the clique if it is connected to every other reference in the current clique. When it is impossible to add another reference to the clique we start the process again creating a new clique. We repeat this clique-building process until all the references are part of a clique. Finally, we treat each resulting clique as a cluster.

5.5.4 Choosing representative references

References within a cluster are guaranteed to be similar to one another but may not be identical. In this final step, we choose a representative reference for each cluster.

Within each cluster we first group together references that share at least 95% of their words and contain exactly the same set of data tuples. The references within each such group are almost identical. We then consider the group containing the largest number of references. If the size of this largest group is greater than three, we select the reference within it that contains the most words as the representative for the cluster – in this case we are essentially selecting a reference that was submitted by three or more workers as the representative. However, if there is a tie in the size of the largest group, we make use of the fact that the references in a group have identical data tuples, and we pick the representative reference from the group containing the largest number of data tuples. This choice is based on the assumption that the group containing fewer data tuples is incomplete and omits one or more relevant tuples. We have found this tie-breaking procedure to work well in practice.

If none of the groups within a cluster contain three or more references, we create a representative reference r for the entire cluster as follows. We set the collection of text phrases for the representative r_t to include all the words that appear in three or more

references in the cluster. Similarly, we set the collection of data tuples for the representative r_d to include all the tuples that appear in three or more references in the cluster. Finally, if a cluster contains fewer than three references we create r by taking the union of both the text phrases and data tuples across all of the references in the cluster. Thus, in both of these cases we synthesize a representative reference that combines information contained in multiple worker generated references.

5.6 Results

To test our pipeline, we gathered a corpus of documents from the Web containing charts. We targeted articles written for a general audience, such as blogs on news websites (the Economist's Graphic Detail [31], the Guardian's DataBlog [41]) and reports from agencies focusing on public policy (Pew Research [77], and a governmental health services agency [101]). We used a variety of sources within this domain to cover different authorship styles. Bar charts are one of the most common forms of charts on these sites as they are familiar to general audiences. We therefore limited our corpus to a set of 18 documents containing only bar charts.

The corpus includes 35 bar charts all together. We used ReVision to extract the marks and data from 20 of these charts and manually extracted the marks and data from the rest. We then manually split the documents into set of 49 paragraph-chart pairs. We withheld 9 of these pairs for use in gold tasks and gathered worker references for the remaining 40. We also asked our experts to manually produce a gold standard reference sets for all 49 pairs using the procedure described earlier. The average distance between the reference sets produced independently by our experts was 0.22, using the distance measure from Chapter 5.4.2.

We then used Amazon's Mechanical Turk to ask 10 crowd workers to extract references for each of the 40 paragraph-chart pairs, yielding a total of 400 reference sets generated by 77 unique crowd workers. Each microtask asked workers to extract references for one paragraph-chart pair and we paid workers $0.15 for completing each such task. The 400 reference sets generated by workers yielded a total of 1127 distinct references, of which 207 were extracted by at least three workers. We then asked another five workers to vote on the correctness of the 920 references that were extracted by fewer than three workers. Each of these workers voted on 10 references and was paid an average of $0.75 per microtask. 263 unique workers completed the rating task, and none of these workers had previously completed the reference extraction task.

5.6.1 Two distances to the gold standard

For some applications, such as our highlighting application, extracting which sentences refer to which tuples may be sufficient, and we can derive these sentence-level references from our computed references. We therefore used two distances to compare the worker results and our computed results to the gold standard: a phrase-level distance and a sentence-level

distance.

To compute phrase-level distance, we used our distance measure for reference sets $d(W, G)$. To compute sentence-level distance, we first split the text into sentences using the Punkt sentence tokenizer in NLTK [9]. We then compute a feature vector for each reference set that contains an entry for each sentence-tuple pair. An entry is set to 1 if the sentence and tuple refer to each other, 0 otherwise. Finally, we compute an F_1 score for the feature vector and convert to a distance by computing $1 - F_1$.

5.6.2 Voting, clustering, and merging improves on the average worker

Of the 400 worker generated reference sets, 220 (55%) passed the accompanying gold check. Figure 5.8 shows the phrase-level distances between each worker's reference set and the gold standard for each of the 40 paragraph-chart conditions, while Figure 5.9 shows the sentence-level distances. Each column in each figure represents one condition and lower points are better because they are closer to the gold standard. The light blue dots indicate workers that did not pass the gold check, while the darker blue circles indicate workers who passed it. Red circles represent the distances between our set of unified references after clustering and merging, but without voting, and the gold standard. Finally, green circles represent the distances between our references after voting, clustering, and merging and the gold standard.

Figures 5.8 and 5.9 also show that there is a relatively large spread in the distances between the worker reference sets and the gold reference sets in every condition. Even when we reject the worker references sets that failed the gold check the spread is large. Using our clustering and merging techniques produces references that are closer to the ground truth than the average of the workers (with or without the gold check). However in most cases the best workers can generate reference sets that are closer to the gold standard (i.e., lower in the chart) than our unified reference sets.

Figure 5.10 (left) aggregates the phrase distances across all 40 conditions. It shows that the average phrase distance between all of the worker generated references and the gold standard is 0.54. Limiting to just the workers who pass the gold check the average distance reduces to 0.47 and when we apply our clustering and merging algorithms the average distance reduces still further to 0.39. Finally, adding voting to our clustering and merging algorithms reduces the average distance to 0.37. Our voting, clustering, and merging improves the reference sets by 32% compared to all workers and 23% compared to the workers who passed the gold check.

Figure 5.10 (right) aggregates the sentence distances across all 40 conditions. As expected, the sentence distances are much lower than the phrase distances. However, we observe a similar trend to the phrase distances. The average sentence distance between all of the worker references to the gold standard is 0.35. Restricting workers to those who passed the gold reduces to the distance to 0.29, and adding clustering and merging further reduces the distance to 0.22. Finally, using worker votes in conjunction with clustering and merging results in an average distance to the gold of 0.20. This represents a 43% improvement over

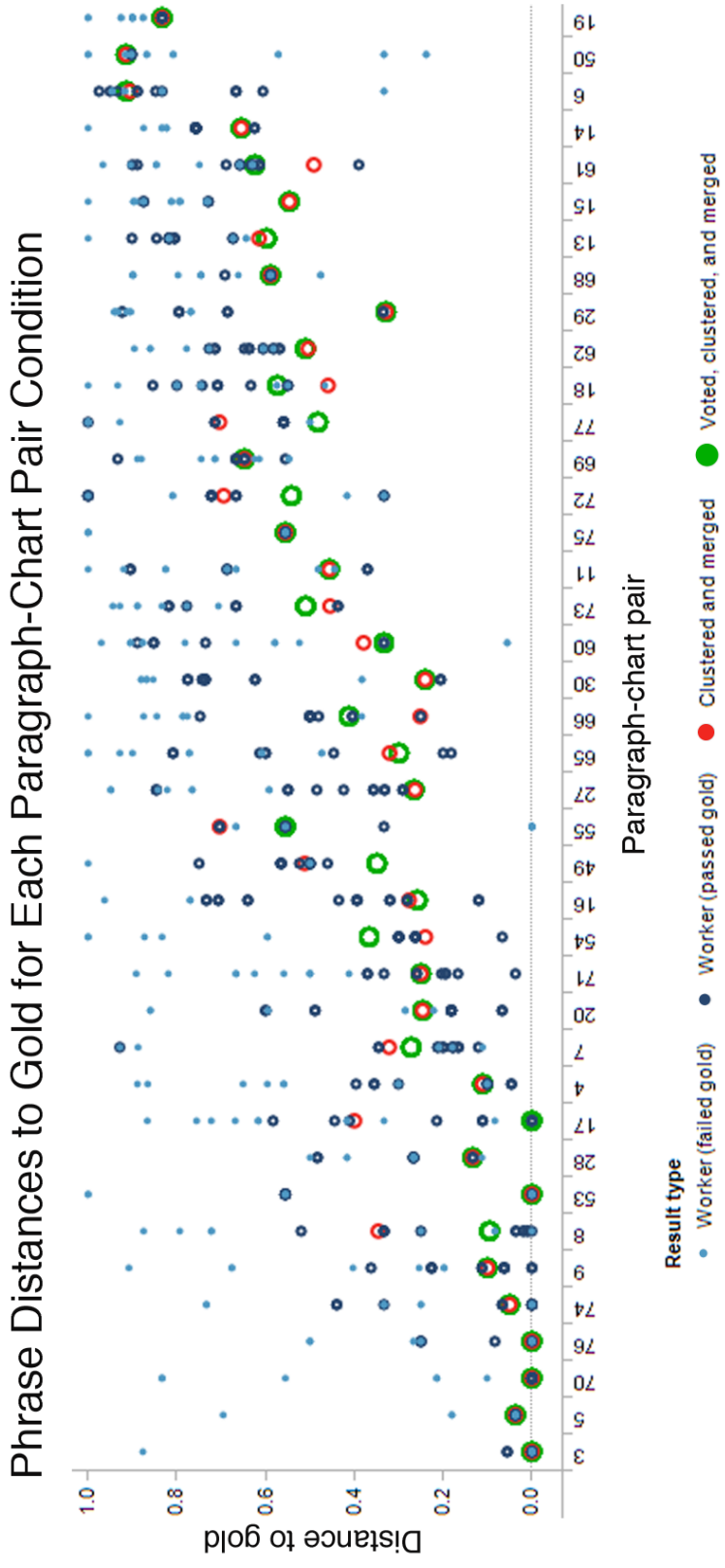


Figure 5.8: Phrase-level distances between the gold standard references and the worker-generated references from all the workers (light blue), workers who passed the gold (dark blue), the clustered and merged references (red), and the clustered and merged references after using worker votes (green). Lower distances are better. Each column represents a paragraph-chart pair condition.

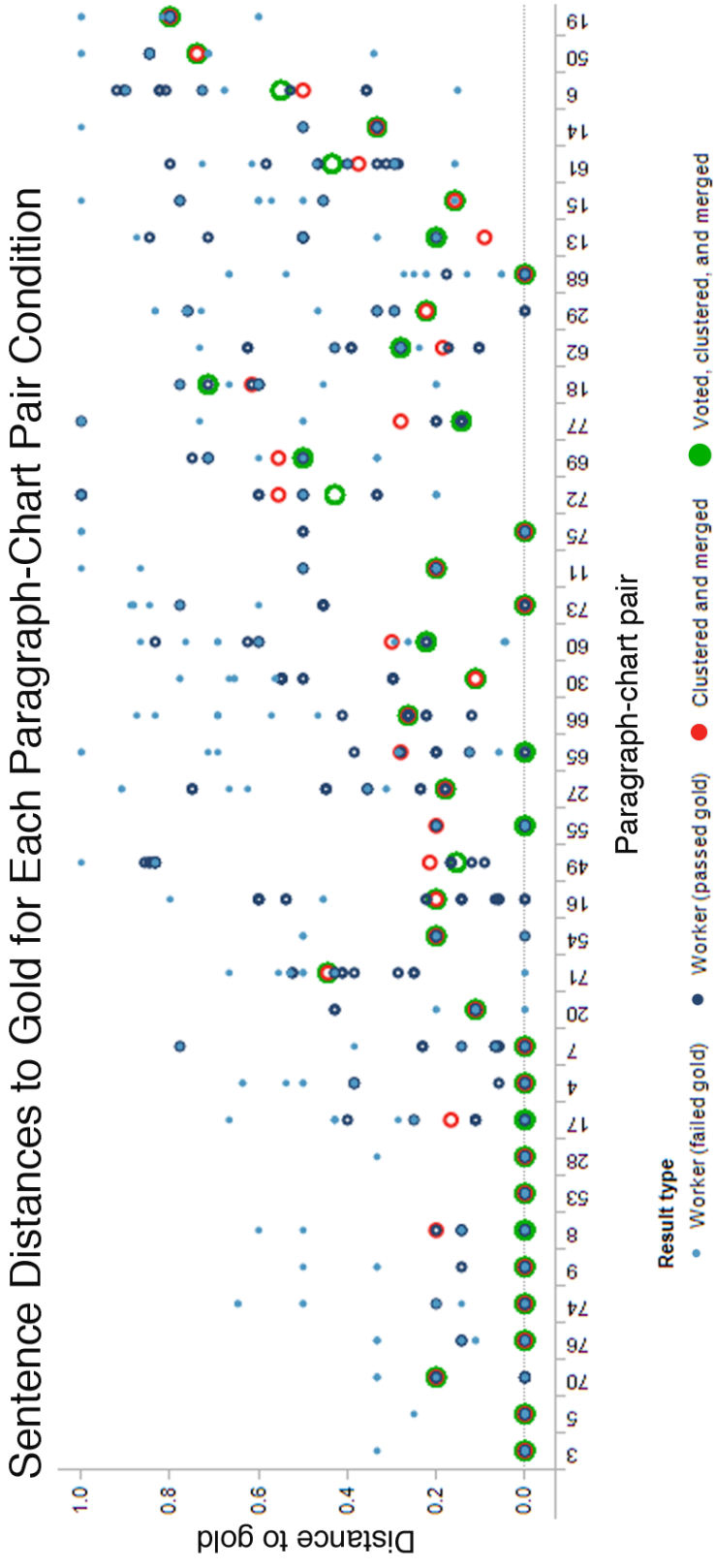


Figure 5.9: Sentence-level distances between the gold standard references and the worker-generated references from all the workers (light blue), workers who passed the gold (dark blue), the clustered and merged references (red), and the clustered and merged references after using worker votes (green). Lower distances are better. Each column represents a paragraph-chart pair condition.

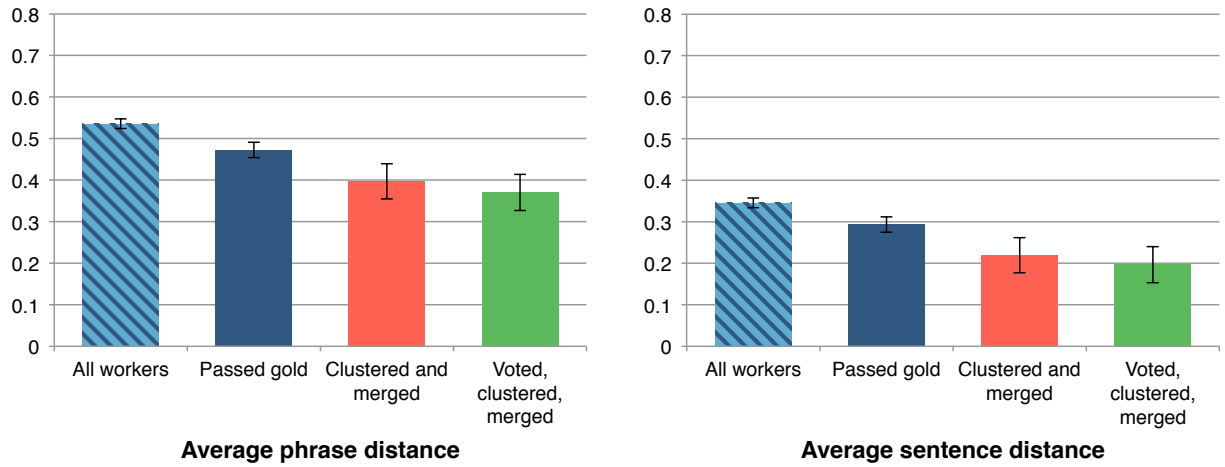


Figure 5.10: Average (left) phrase distances, and (right) sentence distances aggregated across all 40 paragraph-chart pair conditions. Average distances of all worker references (striped bar), references from workers who passed the gold (dark blue), our clustered and merged references without voting (red), and our clustered and merged reference with voting (green) to the gold references. Clustering and merging results in references that are closer to the gold standard than the average worker's set of references, and using worker votes to filter references before clustering and merging further improves the final unified references.

the average worker, and a 33% improvement over the average worker who passed the gold check.

5.7 Application: Interactive Document Viewer

Reading a document and correctly identifying the references between the text and the chart can be difficult (Figure 5.1). We have developed a proof-of-concept interactive document viewing application that uses the references generated by our crowdsourced extraction pipeline to explicitly highlight these correspondences. As shown in Figures 5.1 and 5.11 our implementation presents a document as a paragraph of text and a chart. The user can click and drag to select text and our application highlights the corresponding data-encoding marks in the chart by fading out the surrounding marks.

More specifically we identify all extracted references whose text phrases are fully contained within the selected text. We then look up the data tuples for all such references and highlight the corresponding visual marks in the chart. If we find that there are no references with text phrases that are fully contained within the selected text we use a more lenient highlighting strategy. We identify all of the references whose text contains at least one word of the selected text and then highlight the visual marks for all of the data tuples in the resulting set of references. We implement the mark highlighting as a graphical overlay on

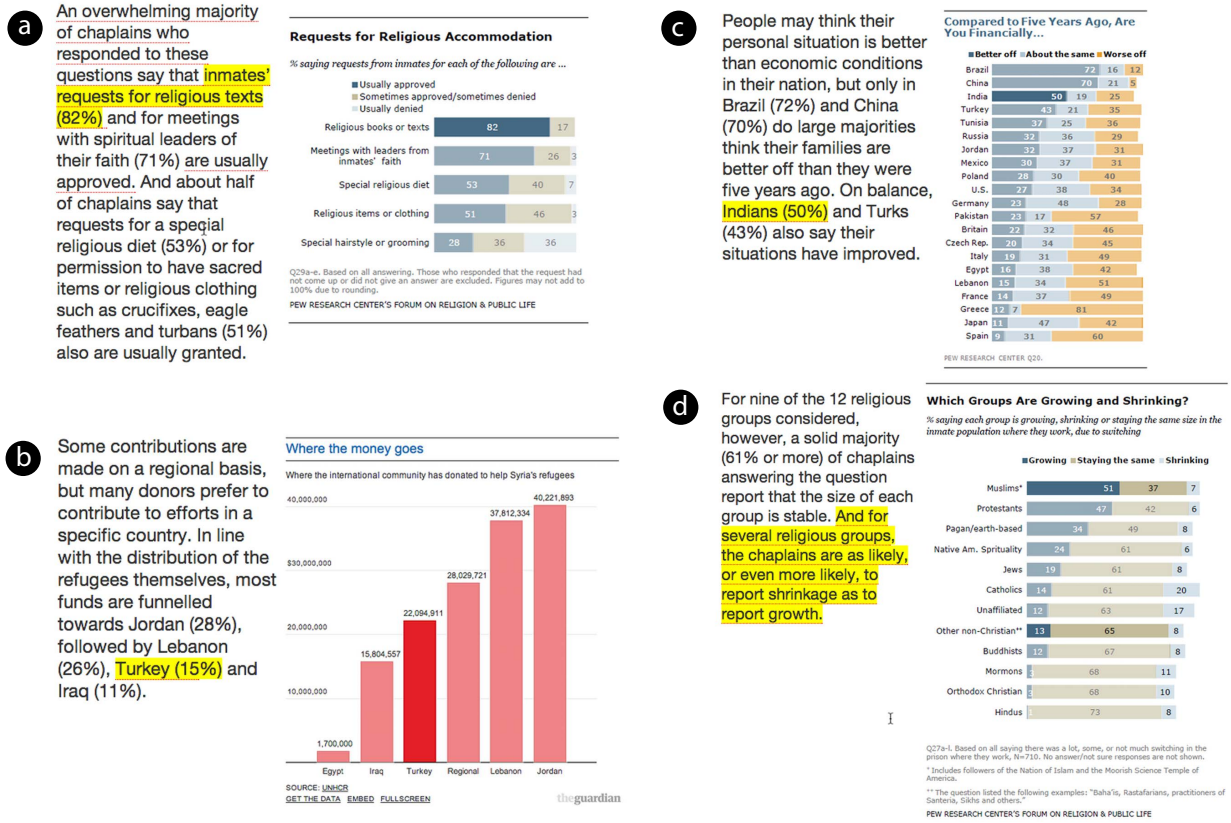


Figure 5.11: Our interactive document viewing application lets users select text (yellow background) and it highlights the corresponding visual marks in the chart (fully saturated bars). The application also places red underlines beneath related phrases. In all four cases the viewer is highlighting marks based on our crowdsourced references after clustering and merging. Examples (a), (b) and (c) show paragraph-chart pairs in which our pipeline creates high quality references. Example (d) shows a paragraph-chart pair for which our pipeline did not extract the correct reference.

the chart bitmap (Chapter 4). We also draw red underlines beneath all of the text phrases in the references that we highlight to further help readers connect the text with the chart.

While Figures 5.1 and 5.11 show our application working with our clustered and merged reference sets, the application can also be used to explore the raw worker generated reference sets, as well as the gold standard reference sets. Our application with access to all of these reference sets is available online at `http://voicebox.eecs.berkeley.edu/textref/apps/highlighting/`.

5.8 Discussion

The large spread of distances in Figure 5.8 between the raw worker generated references and the gold standard suggests that extracting references for some paragraph-chart pairs can be easier than for others.

Examining the raw worker generated references we noticed that workers were most successful when short text phrases in paragraph corresponded with a single data tuple (e.g.. one visual mark) in the chart. Consider the examples in Figure 5.11. In example (a), the text phrase "inmates' request for religious texts (82%)" corresponds to a single blue bar that is labeled with the words *religious books or texts* and the number *82*. Similarly in examples (b) and (c) text phrases "Turkey (15%)" and "Indians (50%)" correspond to bars labeled *Turkey* and *India* respectively. In these cases the simple text phrases and the labels in the chart mark it relatively easy to correctly extract the references. Indeed, these three examples correspond to paragraph-chart conditions 9, 76 and 28 respectively. Looking them up in Figure 5.8 (left) we see that workers who passed the gold check extracted relatively high-quality references that were close to the gold standard (dark blue circles are close to zero).

Example (d) is much more complicated. The text phrase "And for several religious groups, the chaplains are as likely, or even more likely, to report shrinkage as to report growth." refers to the set of bar segments for religions in which shrinkage is larger than or equal to growth. In this case the set of visual marks corresponding to this text include the *growing* and *shrinking* bar segments for *Catholics, Unaffiliated, Mormons, Orthodox Christian* and *Hindus*. Moreover the minimal text phrases for this reference only includes the words "several", "as likely, or even more likely", "shrinkage" and "growth". All the other words in the full sentence are extraneous. This challenging example corresponds to condition 6 and as shown in Figure 5.8 (left) the workers who passed the gold check for this condition produced relatively poor-quality references.

Figure 5.1 also shows a complicated reference for condition 30 in which the text refers to 13 bar segments in the chart. While many of the workers generated mediocre quality references for this condition, our clustering and merging algorithms were able to combine the best information from the workers and extracts a unified set of references that are much better than any of the individual worker generated reference sets. Nevertheless an open direction for future work is to develop techniques to help crowd workers correctly extract references in such complicated cases.

Chapter 6

Future Work

This dissertation has presented techniques for improving the usability of existing visualizations by extracting metadata (i.e., the marks, data, and references to text) and using this metadata to augment the visualizations. There are multiple avenues of future work that this research opens.

6.1 Extending mark and data extraction

In Chapter 3 we presented algorithms for extracting marks and data from bar and pie charts. However, we only considered charts that did not contain textured marks or 3D effects due to gradient shading or perspective. Charts containing these effects are quite common, since software packages like Microsoft Excel offer users such charts as default options (e.g., 3D bar charts or 3D cones). 3D effects have been shown to hamper graphical perception [22; 33; 91], so charts containing these effects are particularly attractive targets for redesign. Extending our algorithms to handle these charts would add further value to the redesign application.

In addition, our extraction algorithms do not parse legends. Legends frequently contain keys to color or shape encodings in the data, as in grouped bar or pie charts, which causes our algorithms to miss labeling information.

Another interesting avenue for future work is designing mark and data extraction algorithms for other chart types, such as bubble charts, line charts, donut charts, and scatterplots. Although our bar and pie chart algorithms cannot be directly applied to new chart types, we posit that some of the techniques that we developed could be reused. For example, the axes in line charts are very similar to the axes in bar charts – they tend to be placed at the left and bottom of the chart. We could therefore use our algorithms for extracting axes in bar charts to extract the axes in line charts. Similarly, we could adapt the use of RANSAC to find the pie in a pie chart to identify bubbles in a bubble chart, the donut in a donut chart, or marks in a scatterplot.

6.2 Improving Graphical Overlays

We could also further extend our overlay taxonomy. Currently, it is focused on bar, pie and line charts. We believe that other types of visualizations such as treemaps or parallel coordinate displays could benefit from the types of overlays we propose in our taxonomy. However the specific designs are likely to differ depending on the chart type.

Our overlay designs are inspired by perceptual and cognitive models of graph comprehension. However, many of these models have not been fully validated through empirical studies. Although some researchers have begun investigating optimal design parameters for gridline overlays [4; 44; 99], the effectiveness of other overlay types is still untested. Our taxonomy provides a framework which could guide future experiments by providing a space of the design parameters. Testing how the design parameters affect the perceptual and cognitive processes involved in chart reading tasks will provide guidelines for overlay creation, as well as potentially reveal missing categories in our taxonomy.

6.3 Improving text reference extraction

We envision multiple possible ways to improve our text reference extraction pipeline. Our crowdsourcing pipeline could be improved to help workers accurately extract complex reference with text phrases that are non-contiguous and refer to multiple data tuples. One possible way to improve our pipeline is to more thoroughly train our workers on our gold standard guidelines. However, we must be careful to avoid presenting workers with complicated prompts, as they increase the cognitive burden on the worker, resulting in lower participation and a higher chance of misunderstanding.

It may also be possible to automatically extract references using a combination of natural language processing, computer vision and machine learning techniques. However our crowdsourcing results suggest that extracting high-quality, minimal references between text and charts is challenging even for humans and doing it automatically is likely to require sophisticated processing techniques. Thus, it may be most fruitful to combine these algorithmic techniques with crowdsourcing or human-in-the-loop intervention.

6.4 Improved and Novel Applications

In this thesis, we have presented three applications that make use of extracted information from charts: a redesign application (Chapter 3), a graphical overlay generator (Chapter 4), and a highlighting application to aid readers of documents containing charts (Chapter 5). However, these applications are proof-of-concept, and there are many improvements that could be made. In addition, there is a large space of unexplored applications that would benefit from extracted marks, data, and references from charts.

6.4.1 Extensions to the redesign application

Chart design often involves semantic data or connotations beyond the raw data tuples. For example, the y-axis labels in the leftmost chart of Figure 3.11 are ordered by increasing educational level. By treating labels as nominal data, our redesign does not respect this ordinal relation. Consequently, our design gallery should support sorting data by both value and the initial observed order. Stylistic choices such as color and typography can also shape the message of a chart; our redesigns do not always retain these messages. However, by extracting the underlying data, our system creates an opportunity for redesign (both automatically and by hand) that would otherwise be prohibitively difficult.

6.4.2 Extensions to the graphical overlays system

There are a number of improvements we could make to our graphical overlays system. It only considers the marks and axes of the chart and may produce overlays that interfere with background graphics, descriptive text, or legends (e.g., Figure 4.9-1). Automated techniques for finding background graphics and text would allow our system produce overlays that do not obscure or overlap with these graphic and text elements.

Our label placement algorithms can generate overlaps that make it difficult to read the label text or the underlying chart elements. Such overlap is especially problematic if the overlay author chooses to label lots of closely spaced marks. More sophisticated label layout algorithms, such as approaches based on simulated annealing [113], could mitigate these issues.

Finally, our overlays system currently provides default design parameters (e.g., highlight color, font type, etc.) for each overlay and chart type. We chose these defaults via ad-hoc experimentation; we tested a few different parameter values across a small corpus of charts and selected those that we thought worked best. Automatic selection of parameter values based on additional analysis of the chart image could improve the default settings. For example, we could use the size of the chart to suggest the number of gridlines based on graphical perception results (e.g., [44]), or we could select a highlight hue that differs strongly from the rest of the chart.

6.4.3 Novel applications for text references

We believe that extracting an accurate set of references for a document opens the door to a variety of applications. Our interactive document viewer lets readers select text and see the corresponding visual marks. One extension is to let readers select marks in the chart and see where they are referenced in the text (Figure 6.1).

The extracted text references could also be used to help authors verify their references in text to charts. For example, an author could input a chart and a paragraph into the

[2] http://www.huffingtonpost.com/2012/06/17/gallup-poll-race-barack-obama_n_1589937.html

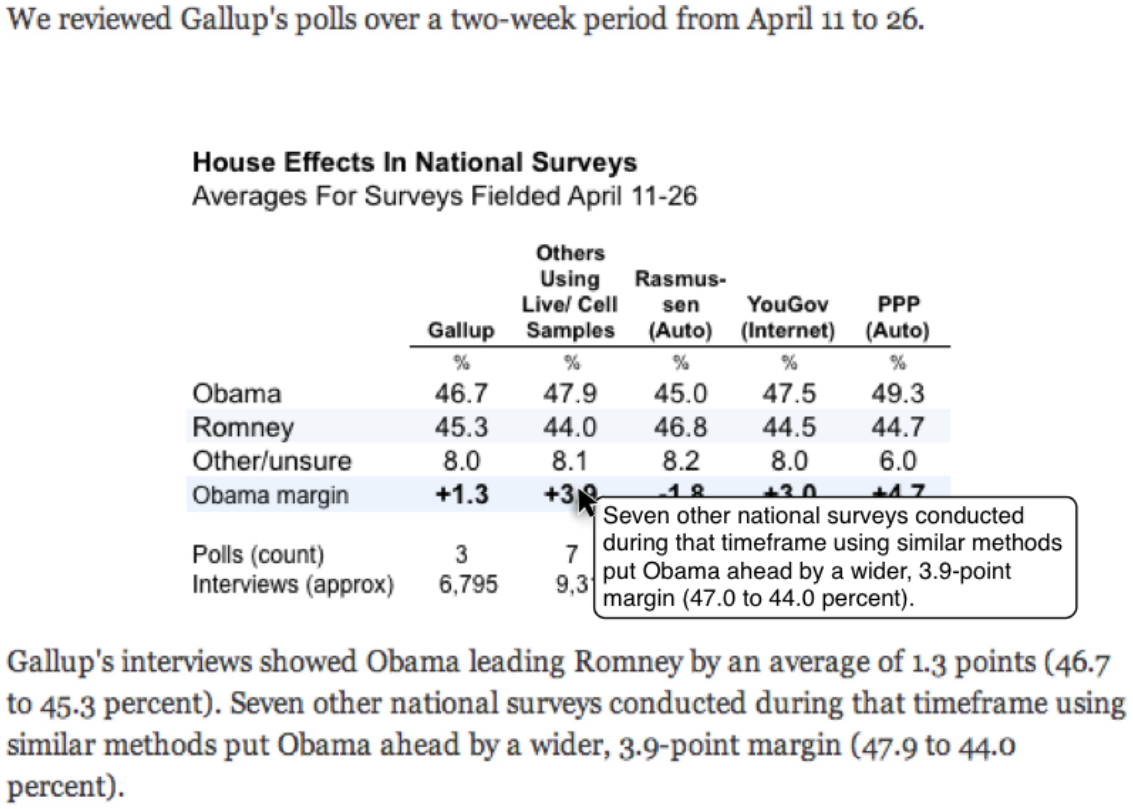


Figure 6.1: Mockup of an extension to the highlighting application from Chapter 5 that overlays related sentences on a chart. (Source: Huffington Post[2])

application, and the application could indicate ambiguous references in the text, or elements in the chart that are not yet referenced in the text.

It may be possible to go further than verification and automatically create snippets of text that refer to an input chart. We could use a set of vetted, gold text references to create a generative model for how authors refer to charts. Such a model could then suggest text to authors that reference parts of the chart.

Finally, having access to the marks, data, and references in charts opens the door to many other applications beyond the ones presented in this thesis. Some examples for using the marks and data include building better screen-readers for charts. For example, a screen reader for charts could pull in the relevant surrounding text as it described the chart. The references could be used to aid document authors: as an author writes text and designs figures such an aid might warn authors if references between the text and charts are missing (e.g. the author wrote about a bar that doesn't appear in the chart). There is also the potential to use text references as metadata for in information retrieval systems. Having the text references might allow for more relevant search results, particularly in returning figures in documents.

Chapter 7

Conclusion

Visualizations abound in printed media and on the web, and the growth of visualization authoring tools means that visualizations will only become more ubiquitous. However, creating effective, usable visualizations remains a challenge, as the designer must consider both the visual encodings and the tasks viewers might wish to perform with the visualization.

In this dissertation, we explored techniques to modify and customize existing visualizations to improve their usability for a wider range of readers. We first presented algorithms for extracting the marks and data from existing raster bar and pie charts that obey some common assumptions – this information is necessary for modifying the visualizations. We tested our algorithms against a corpus of 52 bar charts and 53 pie charts and found that they accurately extracted the marks from 41/52 (79%) of bar charts and 33/53 (62%) of pie charts. From these charts, our algorithms successfully extracted the data from 29/41 (71%) of bar charts and 21/33 (64%) of pie charts. We then showed how the marks and data can be used for chart redesign via an application that shows multiple alternate redesigns with customizable styles.

Next, we considered how to augment existing charts without changing their visual encodings. We introduced graphical overlays to enable customization of charts to support specific perceptual and cognitive processes that viewers undergo when reading a chart. Using graphical overlays, chart authors can directly support specific viewer tasks, or introduce interactivity to allow the users to create their own overlays. We presented a taxonomy of five categories of overlays and a system that produces these overlays using marks and (in some cases) data extracted from charts. Finally, we showed a system that allows authors to add all five types of overlays to existing charts.

Finally, we investigated how to surface the references between a chart and related text. We described a crowdsourcing pipeline that can extract such references, and presented distance measures to compare these references to a gold standard. We then showed how such references can be used to facilitate reading with a proof-of-concept application for interactive document viewing. As more and more reading is done on electronic devices such interactive viewing can greatly improve the reading experience.

As we build upon our understanding of graphical perception and cognitive processes, we

better learn how to effectively design visualizations. In this dissertation, we have laid the groundwork for retroactively applying new design guidelines to existing charts, as well as developed tools to make future visualizations more useful to a wider variety of viewers.

Bibliography

[1] Ahmad, S., Battle, A., Malkani, Z., and Kamvar, S. The jabberwocky programming environment for structured social computing. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, ACM (New York, NY, USA, 2011), 53–64.

[2] Arizona Software. GraphClick. `http://www.arizona-software.ch/ graphclick`. Retrieved Jun 11, 2012.

[3] Bargeron, D., and Moscovich, T. Reflowing digital ink annotations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, ACM (New York, NY, USA, 2003), 385–393.

[4] Bartram, L., Cheung, B., and Stone, M. The effect of colour and transparency on the perception of overlaid grids. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (dec. 2011), 1942 –1948.

[5] Baudisch, P., and Rosenholtz, R. Halo: a technique for visualizing off-screen objects. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, ACM (New York, NY, USA, 2003), 481–488.

[6] Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, ACM (New York, NY, USA, 2010), 313–322.

[7] Bernstein, M. S., Teevan, J., Dumais, S., Liebling, D., and Horvitz, E. Direct answers for search queries in the long tail. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 237–246.

[8] Bertin, J. *Sémiologie Graphique*. Gauthier-Villars, 1967.

[9] Bird, S. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, Association for Computational Linguistics (Stroudsburg, PA, USA, 2006), 69–72.

[10] Bormann, I. DigitizeIt. `http://www.digitizeit.de`. Retrieved Jun 11, 2012.

[11] Bostock, M., and Heer, J. Protovis: A graphical toolkit for visualization. *IEEE Trans Visualization & Comp Graphics 15*, 6 (2009), 1121–1128.

[12] Bostock, M., Ogievetsky, V., and Heer, J. D3: Data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (dec. 2011), 2301 –2309.

[13] British Broadcasting Corporation. EU budget plans for 2011.

http://www.bbc.co.uk/news/uk-politics-11645975. Retrieved Mar 31, 2012.

[14] Cairo, A. *The Functional Art: An introduction to information graphics and visualization*. New Riders, 2012.

[15] Callison-Burch, C., and Dredze, M. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, Association for Computational Linguistics (Stroudsburg, PA, USA, 2010), 1–12.

[16] Captricity. http://www.captricity.com. Retrieved Oct 20, 2013.

[17] Card, S. K., Mackinlay, J. D., and Shneiderman, B., Eds. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[18] Carpenter, P. A., and Shah, P. A model of the perceptual and conceptual processes in graph comprehension. *Journal of Experimental Psychology Applied 4*, 2 (1998), 75–100.

[19] Chen, D., Odobez, J., and Bourlard, H. Text detection and recognition in images and video frames. *Pattern Recognition 37*, 3 (2004), 595–608.

[20] Chilton, L. B., Little, G., Edge, D., Weld, D. S., and Landay, J. A. Cascade: crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 1999–2008.

[21] Christopher D. Manning, Prabhakar Raghavan, H. S. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[22] Cleveland, W. S. *Visualizing Data*. Hobart Press, 1993.

[23] Cleveland, W. S. *The Elements of Graphing Data*, 2nd ed. Hobart Press, Oct. 1994.

[24] Cleveland, W. S., Harris, C. S., and McGill, R. Judgments of Circle Sizes on Statistical Maps. *Journal of the American Statistical Association 77*, 379 (1982), 541–547.

[25] Cleveland, W. S., and McGill, R. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association 79*, 387 (1984), 531–554.

[26] Cleveland, W. S., and McGill, R. Graphical Perception and Graphical Methods for Analyzing Scientific Data. *Science 229*, 4716 (1985), 828–833.

[27] Coates, A., Lee, H., and Ng, A. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. *Advances in Neural Information Processing Systems* (2010).

[28] Collins, C., Penn, G., and Carpendale, S. Bubble sets: Revealing set relations with isocontours over existing visualizations. *Visualization and Computer Graphics, IEEE Transactions on 15*, 6 (nov.-dec. 2009), 1009 –1016.

[29] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning 20* (1995), 273–297.

[30] CrowdFlower. http://www.crowdflower.com. Retrieved Oct 20, 2013.

[31] Economist Graphic Detail. http://www.economist.com/blogs/graphicdetail. Retrieved Sep 17, 2013.

[32] Fekete, J.-D., Wang, D., Dang, N., Aris, A., and Plaisant, C. *Overlaying graph links*

on treemaps, vol. pages. Citeseer, 2003, 2–3.

[33] Few, S. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press, Berkeley, CA, 2004.

[34] Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24* (June 1981), 381–395.

[35] Fitzgibbon, A., Pilu, M., and Fisher, R. Direct least square fitting of ellipses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 21*, 5 (May 1999), 476 –480.

[36] Furnas, G. W. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '86, ACM (New York, NY, USA, 1986), 16–23.

[37] Garner, S. R. Weka: The waikato environment for knowledge analysis. In *In Proc. of the New Zealand Computer Science Research Students Conference* (1995), 57–64.

[38] Gleicher, M., Correll, M., Nothelfer, C., and Franconeri, S. Perception of average value in multiclass scatterplots. *Visualization and Computer Graphics, IEEE Transactions on 19*, 12 (2013), 2316–2325.

[39] Google Charts. `https://developers.google.com/chart/`. Retrieved Oct 14, 2013.

[40] Google Finance. `http://www.google.com/finance/`. Retrieved Oct 23, 2013.

[41] Guardian DataBlog. http://www.theguardian.com/datablog. Retrieved Sep 17, 2013.

[42] Gustafson, S., Baudisch, P., Gutwin, C., and Irani, P. Wedge: clutter-free visualization of off-screen locations. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, ACM (New York, NY, USA, 2008), 787–796.

[43] Harrower, M., and Brewer, C. Colorbrewer.org: an online tool for selecting colour schemes for maps. *The Cartographic Journal 40*, 1 (2003), 27–37.

[44] Heer, J., and Bostock, M. Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design. In *ACM CHI* (2010), 203–212.

[45] Heer, J., Kong, N., and Agrawala, M. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems* (2009), 1303–1312.

[46] Heer, J., and Stone, M. Color naming models for color selection, image editing and palette design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 1007–1016.

[47] Heer, J., Viégas, F. B., and Wattenberg, M. Voyagers and voyeurs: supporting asynchronous collaborative information visualization. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 2007), 1029–1038.

[48] Hill, W. C., and Hollan, J. D. Deixis and the future of visualization excellence. In *Proc. of IEEE Visualization* (1991), 314–320, 431.

[49] Hsueh, P.-Y., Melville, P., and Sindhwani, V. Data quality from crowdsourcing: a

study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, HLT '09, Association for Computational Linguistics (Stroudsburg, PA, USA, 2009), 27–35.

[50] Huang, W., and Tan, C. L. A system for understanding imaged infographics and its applications. In *Proceedings of the 2007 ACM symposium on Document engineering*, DocEng '07, ACM (New York, NY, USA, 2007), 9–18.

[51] Huang, W., Tan, C. L., and Leow, W. K. Model-based chart image recognition. In *Graphics Recognition*, J. Lladós and Y.-B. Kwon, Eds., vol. 3088 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, 87–99.

[52] Hullman, J., Adar, E., and Shah, P. The impact of social information on visual judgments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 1461–1470.

[53] Hullman, J., and Diakopoulos, N. Visualization rhetoric: Framing effects in narrative visualization. *IEEE TVCG 17*, 12 (2011), 2231–2240.

[54] Hullman, J., Diakopoulos, N., and Adar, E. Contextifier: automatic generation of annotated stock visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 2707–2716.

[55] Javed, W., McDonnel, B., and Elmqvist, N. Graphical Perception of Multiple Time Series. *IEEE Transactions on Visualization and Computer Graphics* (2010).

[56] Kandogan, E. Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations. In *IEEE VAST* (2012), 73–82.

[57] Kittur, A., Smus, B., Khamkar, S., and Kraut, R. E. Crowdforge: crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, ACM (New York, NY, USA, 2011), 43–52.

[58] Kong, N., and Agrawala, M. Perceptual interpretation of ink annotations on line charts. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, ACM (New York, NY, USA, 2009), 233–236.

[59] Kong, N., and Agrawala, M. Graphical Overlays: Using Layered Elements to Aid Chart Reading. *IEEE Transactions on Visualization and Computer Graphics* (2012).

[60] Kong, N., Heer, J., and Agrawala, M. Perceptual guidelines for creating rectangular treemaps. *IEEE Transactions on Visualization and Computer Graphics 16* (2010), 990–998.

[61] Kosslyn, S. M. Understanding charts and graphs. *Applied Cognitive Psychology 3*, 3 (1989), 185–225.

[62] Kriz, S., and Hegarty, M. Top-down and bottom-up influences on learning from animations. *International Journal of Human-Computer Studies 65*, 11 (2007), 911 – 930.

[63] Kulkarni, A., Can, M., and Hartmann, B. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, CSCW '12, ACM (New York, NY, USA, 2012), 1003–1012.

[64] Lanman, D. Bilateral Filtering. `http://www.mathworks.com/matlabcentral/fileexchange/12191-bilateral-filtering`. Retrieved Oct 25, 2013.

[65] Le, J., Edmonds, A., Hester, V., and Biewald, L. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR 2010 workshop on crowdsourcing for search evaluation* (2010), 21–26.

[66] Lin, S., Fortuna, J., Kulkarni, C., Stone, M., and Heer, J. Selecting semantically-resonant colors for data visualization. *Computer Graphics Forum (Proc. EuroVis)* (2013).

[67] Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, ACM (New York, NY, USA, 2010), 57–66.

[68] Liu, R., Huang, W., and Tan, C. L. Extraction of vectorized graphical information from scientific chart images. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 1 (2007), 521 –525.

[69] Lohse, G. L. A cognitive model for understanding graphical perception. *Human-Computer Interaction 8*, 4 (1993), 353–388.

[70] M. Migurski, T. Carden, and E. Rodenback. Oakland Crimespotting. `http://oakland.crimespotting.org/`. Retrieved Oct 23, 2013.

[71] Mackinlay, J. Automating the design of graphical presentations of relational information. *ACM Trans. Graph. 5*, 2 (1986), 110–141.

[72] Mackinlay, J., Hanrahan, P., and Stolte, C. Show me: Automatic presentation for visual analysis. *Visualization and Computer Graphics, IEEE Transactions on 13*, 6 (2007), 1137 –1144.

[73] Mayfield, J., Lawrie, D., McNamee, P., and Oard, D. Building a cross-language entity linking collection in twenty-one languages. In *Multilingual and Multimodal Information Access Evaluation*, P. Forner, J. Gonzalo, J. Kekäläinen, M. Lalmas, and M. de Rijke, Eds., vol. 6941 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2011, 3–13.

[74] Mobileworks. `http://www.mobileworks.com`. Retrieved Oct 20, 2013.

[75] Nakov, P. Noun compound interpretation using paraphrasing verbs: Feasibility study. In *Artificial Intelligence: Methodology, Systems, and Applications*, D. Dochev, M. Pistore, and P. Traverso, Eds., vol. 5253 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, 103–117.

[76] Oleson, D., Sorokin, A., Laughlin, G., Hester, V., Le, J., and Biewald, L. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. *Proc. HComp* (2011).

[77] Pew Research. `http://www.pewresearch.org/`. Retrieved Sep 17, 2013.

[78] Pinker, S. *A theory of graph comprehension.* Lawrence Erlbaum Associates, 1990, 73–126.

[79] Playfair, W. *Playfair's Commercial and Political Atlas and Statistical Breviary.* Cambridge University Press, 2005.

[80] Prasad, V., Siddiquie, B., Golbeck, J., and Davis, L. Classifying Computer Generated Charts. In *Content-Based Multimedia Indexing Workshop*, IEEE (2007), 85–92.

[81] Rao, R., and Card, S. K. The table lens: merging graphical and symbolic repre-

sentations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, ACM (New York, NY, USA, 1994), 318–322.

[82] Ratwani, R. M., Trafton, J. G., and Boehm-Davis, D. A. Thinking graphically: Connecting vision and cognition during graph comprehension. *Journal of Experimental Psychology: Applied 14*, 1 (2008), 36–49.

[83] Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., and Heer, J. Revision: automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, ACM (New York, NY, USA, 2011), 393–402.

[84] Segel, E., and Heer, J. Narrative visualization: Telling stories with data. *IEEE TVCG 16*, 6 (2010), 1139–1148.

[85] Shneiderman, B. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph. 11*, 1 (1992), 92–99.

[86] Shneiderman, B. The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on* (1996), 336–343.

[87] Simkin, D., and Hastie, R. An information-processing analysis of graph perception. *Journal of the American Statistical Association 82*, 398 (1987), 454–465.

[88] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, Association for Computational Linguistics (Stroudsburg, PA, USA, 2008), 254–263.

[89] Spence, I. The apparent and effective dimensionality of representations of objects. *Human Factors: The Journal of the Human Factors and Ergonomics Society 46*, 4 (2004), 738–747.

[90] Spence, I. Playfair, William (1759–1823). In *Oxford Dictionary of National Biography*. Oxford University Press, 2004.

[91] Spence, I. The Apparent and Effective Dimensionality of Representations of Objects. *Human Factors: The Journal of the Human Factors and Ergonomics Society 46*, 4 (2004), 738–747.

[92] Spotfire. `http://spotfire.tibco.com/`. Retrieved Oct 23, 2013.

[93] Steinberger, M., Waldner, M., Streit, M., Lex, A., and Schmalstieg, D. Context-preserving visual links. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (dec. 2011), 2249 –2258.

[94] Stevens, S. S. On the psychophysical law. *Psychophysical Review 64*, 3 (1957), 153–181.

[95] Stevens, S. S. The psychophysics of sensory function. *American Scientist 48* (1960), 226–253.

[96] Stevens, S. S., and Miguelina, G. Subjective scaling of length and area and the matching of length to loudness and brightness. *Journal of Experimental Psychology 66*, 2 (Aug 1963), 177–186.

[97] Stolte, C., Tang, D., and Hanrahan, P. Polaris: a system for query, analysis, and

visualization of multidimensional relational databases. *Visualization and Computer Graphics, IEEE Transactions on 8*, 1 (2002), 52 –65.

[98] Stone, M. *A Field Guide to Digital Color*. A. K. Peters, 2003.

[99] Stone, M., and Bartram, L. Alpha, contrast and the perception of visual metadata. *Color Imaging Conference* (2009).

[100] Stone, M. C. Color in information display principles, perception, and models. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, ACM (New York, NY, USA, 2004), 21–.

[101] Substance Abuse and Mental Health Services Administration. http://www.samhsa.gov/. Retrieved Sep 17, 2013.

[102] Sun, A., Valentino-DeVries, J., and Seward, Z. A Week on Foursquare. http://graphicsweb.wsj.com/documents/FOURSQUAREWEEK1104. Retrieved Oct 23, 2013.

[103] Tableau. http://www.tableausoftware.com/. Retrieved Oct 23, 2013.

[104] Tableau. Tableau Public. http://www.tableausoftware.com/public/. Retrieved Mar 28, 2013.

[105] Tak, S., and Cockburn, A. Enhanced spatial stability with hilbert and moore treemaps. *Visualization and Computer Graphics, IEEE Transactions on 19*, 1 (2013), 141–148.

[106] Teghtsoonian, M. The judgment of size. *The American Journal of Psychology 78*, 3 (1965), 392–402.

[107] Tomasi, C., and Manduchi, R. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on* (Jan. 1998), 839 –846.

[108] Trickett, S., and Trafton, J. Toward a comprehensive model of graph comprehension: Making the case for spatial cognition. In *Diagrammatic Representation and Inference*, D. Barker-Plummer, R. Cox, and N. Swoboda, Eds., vol. 4045 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, 286–300.

[109] Tufte, E. R. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[110] Tufte, E. R. *Envisioning Information*. Graphics Press, 1990.

[111] Tummers, B. DataThief III. http://www.datathief.org. Retrieved Mar 31, 2012.

[112] UK Department of Energy and Climate Change. National Heat Map. http://tools.decc.gov.uk/nationalheatmap/. Retrieved Oct 23, 2013.

[113] Vollick, I., Vogel, D., Agrawala, M., and Hertzmann, A. Specifying label layout style by example. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology* (2007), 221–230.

[114] Wertheimer, M. Laws of organization in perceptual forms. *A source book of Gestalt psychology* (1938), 71–88.

[115] Willett, W., Ginosar, S., Steinitz, A., Hartmann, B., and Agrawala, M. Identifying Redundancy and Exposing Provenance in Crowdsourced Data Analysis. *IEEE Transactions on Visualization and Computer Graphics* (2013).

[116] Willett, W., Heer, J., and Agrawala, M. Strategies for crowdsourcing social data analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 227–236.

[117] Willett, W., Heer, J., Hellerstein, J., and Agrawala, M. Commentspace: structured

support for collaborative visual analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 3131–3140.

[118] Yang, L., Huang, W., and Tan, C. Semi-automatic ground truth generation for chart image recognition. In *Document Analysis Systems VII*, H. Bunke and A. Spitz, Eds., vol. 3872 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, 324–335.

[119] Yokokura, N., and Watanabe, T. Layout-based approach for extracting constructive elements of bar-charts. In *Graphics Recognition Algorithms and Systems*, K. Tombre and A. Chhabra, Eds., vol. 1389 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1998, 163–174.

[120] Zacks, J., and Tversky, B. Bars and lines: A study of graphic communication. *Memory & Cognition 27*, 6 (1999), 1073–1079.

[121] Zhou, Y. P., and Tan, C. L. Hough technique for bar charts detection and recognition in document images. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 2 (2000), 605 –608 vol.2.

[122] Ziemkiewicz, C., and Kosara, R. The shaping of information by visual metaphors. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov/Dec 2008), 1269–1276.