Pay “Attention” to Chart Images for What You Read on Text

Chenyu Yang
Renmin University of China, China
yangchenyu@ruc.edu.cn

Ruixue Fan
Renmin University of China, China
fanruixue@ruc.edu.cn

Nan Tang
QCRI, HBKU / HKUST (GZ)
Qatar / China
ntang@hbku.edu.qa

Meihui Zhang
Beijing Institute of Technology, China
meihui_zhang@bit.edu.cn

Xiaoman Zhao*
Renmin University of China, China
xiaomanzhao@ruc.edu.cn

Ju Fan
Renmin University of China, China
fanj@ruc.edu.cn

Xiaoyong Du
Renmin University of China, China
duyong@ruc.edu.cn

ABSTRACT

Data visualization is changing how we understand data, by showing why’s, how’s, and what’s behind important patterns/trends in almost every corner of the world, such as in academic papers, news articles, financial reports, etc. However, along with the increasing complexity and richness of data visualization, given a text description (*e.g.,* “fewer teens say they attended school completely online (8%)”), it becomes harder for users to pinpoint where to pay attention to on a chart (*e.g.,* a grouped bar chart).

In this demonstration paper, we present a system HiChart for text-chart image highlighting: when a user selects a span of text, HiChart automatically analyzes the chart image (*e.g.,* a jpeg or a png file) and highlights the parts that are relevant to the span. From a technical perspective, HiChart devises the following techniques. *Reverse-engineering visualizations:* given a chart image, HiChart uses computer vision techniques to generate a visualization specification using Vega-Lite language, as well as the underlying dataset; *Visualization calibration by data tuning:* HiChart calibrates the re-generated chart by tuning the recovered dataset through value perturbation; and *Chart highlighting for a span:* HiChart maps the span to corresponding data cells and uses the built-in highlighting functions of Vega-Lite to highlight the chart.

CCS CONCEPTS

• **Human-centered computing → Visualization**.

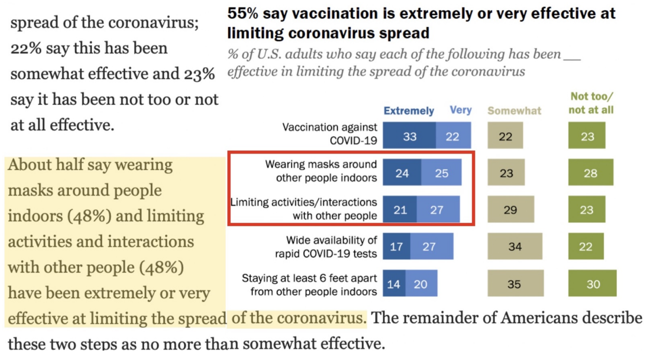
KEYWORDS

chart highlighting, data visualization, data extraction

1 INTRODUCTION

Data visualization is critical to analyze and understand massive amounts of information. Along with the visualizations containing richer information, it becomes harder for users to pay attention

*Xiaoman Zhao is the corresponding author.


spread of the coronavirus; 22% say this has been somewhat effective and 23% say it has been not too or not at all effective.

About half say wearing masks around people indoors (48%) and limiting activities and interactions with other people (48%) have been extremely or very effective at limiting the spread of the coronavirus. The remainder of Americans describe these two steps as no more than somewhat effective.

55% say vaccination is extremely or very effective at limiting coronavirus spread
% of U.S. adults who say each of the following has been ___ effective in limiting the spread of the coronavirus
Extremely
Very
Somewhat
Not too/ not at all
Vaccination against COVID-19
33
22
22
23
Wearing masks around other people indoors
24
25
23
28
Limiting activities/interactions with other people
21
27
29
23
Wide availability of rapid COVID-19 tests
17
27
34
22
Staying at least 6 feet apart from other people indoors
14
20
35
30

Figure 1: HiChart highlights a chart *w.r.t.* the user-selected text for data journalism from the PEW Research.

to specific parts of a visualization that correspond to a given text description. Thus, from an application perspective, inspired by the important text-image application (*e.g.,* if the text mentions “a cat”, then it automatically draws a bounding box of the cat object in the image), this paper studies how to highlight a specific part of data in a chart, to draw users’ attention to the important part of the chart.

Example 1 (**Motivating Example**). *Figures 1 shows a screenshot of news from the PEW research center (https://www.pewresearch.org/). If a reader wants to find the related part of a chart when reading the sentence, the reader needs to consider all the elements of the images at the same time, move attention between image and text to find the corresponding data. Obviously, this would hinder the pace of reading, especially when the chart contains a large amount of information.*

*Specifically, the two numbers (*i.e.*, “48%”) in the selected sentence cannot be directly found in the chart, as they refer to the two blue columns corresponding to “Extreme” and “Very” in the chart. These two columns represent the percentages of people saying a mean is extremely or very effective at limiting the spread of the coronavirus. For example, for the second “48%”, we need to add the parts “Extremely” and “Very” together to obtain the number. For the first “48%”, we cannot get the same number as the data in the chart is rounded to some extent. The examples show that it is time-consuming to find the corresponding part in the chart based on the text, especially for non-expert users.*

To summarize, the users will encounter the following difficulties: (C1) They are hard to locate relevant information from complex visualization charts. (C2) In some cases, the values in the text may be the results of aggregations so they do not even appear in the chart. (C3) Due to errors or rounding of data, the number in the text may not correspond exactly to those shown in the chart.

Existing solutions and their limitations. Chart highlighting solutions have been adopted to address the above difficulties, but they are mainly for pre-defined (or hard-coded) cases. For example, there are some built-in highlight functions in existing visualization tools such as ExcelTV and Tableau which also support chart highlighting given chart data. ExcelTV allows users to highlight the chart by setting criteria in tabular data, such as highlighting "maximum" data items. Tableau allows users to highlight data by defining source and target tables as well as the fields they want to highlight. However, both of them cannot support auto-highlighting based on *text description*. Moreover, using text-image matching methods is not ideal for our problem, because the computer vision models (*e.g.,* CNN [2]) are good at recognizing physical objects (*e.g.,* cats or bikes) instead of data visualizations elements. There are also techniques that can map text to table cells from the natural language processing community [1]. We will integrate these techniques to serve as the solutions of one component of our pipeline for text-chart image highlighting.

Solution overview. In this paper, we consider a different but very common setting: given a chart image where neither the visualization nor the dataset for generating this chart is provided, and a text description that is either user written or a user selected span, our problem is to highlight the part of the chart image for the text specification, as shown in Figure 1. To solve the problem, we propose a novel system, called HiChart, for highlighting a chart image based on a text, which consists of the following components:

- *Reverse-engineering visualizations:* given a chart image, HiChart uses computer vision techniques to generate the visualization specification using Vega-Lite language, as well as the underlying dataset (Section 2.1).
- *Visualization calibration by data tuning:* HiChart calibrates the re-generated chart by tuning the recovered dataset through value perturbation, similar to the forward-propagation concept in machine learning (Section 2.2).
- *Chart highlighting for text:* HiChart maps the text to corresponding data cells and uses the built-in highlighting functions of Vega-Lite to highlight the chart (Section 2.3).

We present a web application to users, and demonstrate HiChart in a typical chart highlighting scenario for digital newspapers in Section 3. We allow users to easily explore news in a variety of topics by selecting a text span (*e.g.,* a sentence or a paragraph) one wants to inspect. We also demonstrate that HiChart can accurately highlight the parts of a visualization chart, which are relevant to the text span provided by the users.

2 OVERVIEW OF HICHART

A high-level overview of HiChart is shown in Figure 2. Given a chart image (*e.g.,* a jpeg or png file) and a span of text as input, HiChart aims to highlight (*e.g.,* using bounding boxes) the part of the image that relates to the input text, as illustrated in Figure 1. To achieve this, HiChart consists of the following three components.

Reverse-engineering visualization. Given a chart image V, this component reverse-engineers the visualization process to generate a visualization specification S (or specification for short) such that chart image V can be re-generated by running specification S. Note that the specification S contains a dataset D, *e.g.,* a table of numerical values. The key challenge is that, as both S and D are not given in advance, we need to *reverse-engineer* the visualization process to obtain them. In the current version of HiChart, we consider Vega-Lite[1] as the language for visualization specification, which is the most popular language in the visualization community.

Visualization calibration by data tuning. Given the reverse-engineered specification S which contains dataset D, the component aims to calibrate D to a refined dataset D' such that the generated visualization chart $S(D')$ is closer to the original image V. To this end, we devise a distance function to quantify the distance between image V and each generated image $S(D')$, *i.e.,* $\text{dist}(S(D'), V)$. Then, the problem of *visualization calibration by data tuning* can be solved by performing perturbation on D to obtain a D' that minimizes the distance $\text{dist}(S(D'), V)$.

Chart highlighting for a text span. Given a span of text T input by a user, the component highlights the particular parts in the chart $S(D')$ (*e.g.,* some bars or lines) that are relevant to T. When the used visualization specification language supports highlighting functionalities, this component is essential to *augment* specification S to S^* by adding the code for highlighting. The key challenge is to find a method to capture the information in the natural language sentence and find the corresponding data in the table so that our system can highlight the chart based on the data.

2.1 Reverse Engineering Visualization

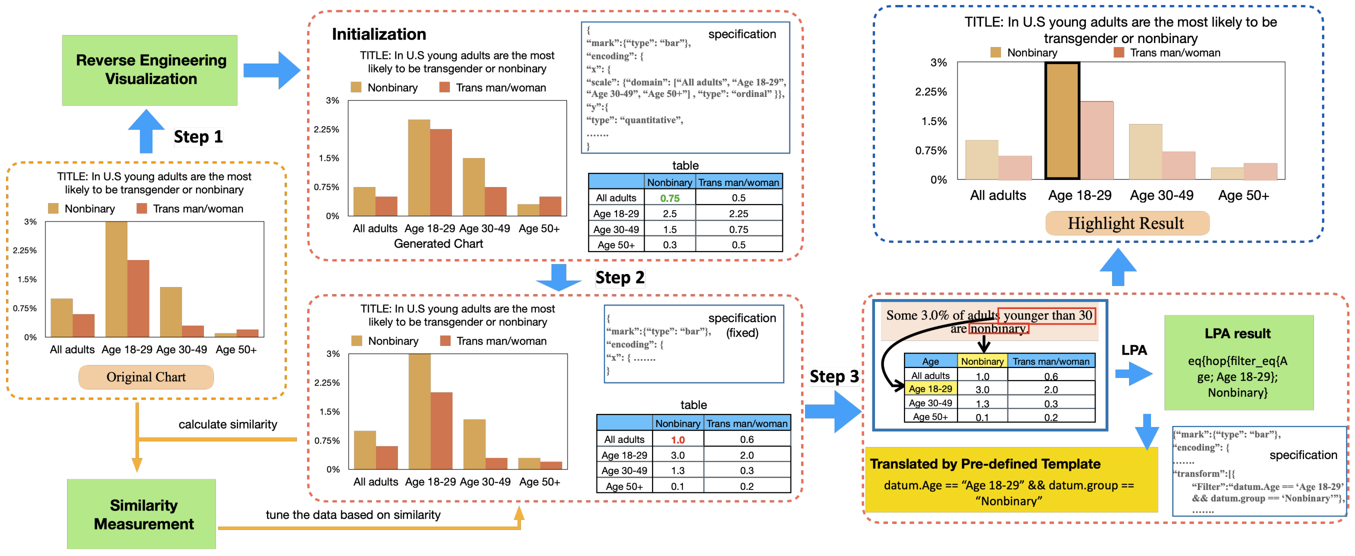
Reverse engineering visualization aims to recover a visual encoding specification S with a dataset D from a chart image V, which is opposite to visualization generation. One limitation of the existing studies on reverse engineering visualization is that they cannot jointly extract the textual elements (*e.g.,* axis labels and tick labels) and numerical data values (*e.g.,* values indicated by the bars) [3, 4]. Recently, some studies [7] focus on recovering both S and D from images, but they are limited to only bar charts. This is because, in practice, it is non-trivial to recover both visualization specification S and the used dataset D from a chart image.

To address this, we develop a reverse-engineering algorithm by extending ChartOCR [4]. Specifically, given a chart image, we use the methods in ChartOCR [4] to obtain key points and text in a chart. With both of them, we can infer textual information in a chart. For dataset D, since ChartOCR can detect key points, we can use the coordinates of textual elements and these key points to calculate numerical values, and match them with the x-axis and the legend labels. Finally, we combine the numerical values and the textual elements to generate a Vega-Lite specification.

2.2 Visualization Calibration by Data Tuning

Extracting accurate values in the dataset D is indispensable for chart highlighting since we cannot find corresponding data with the selected text if the data is inaccurate. However, the previous reverse-engineering visualization process may not be perfect at extracting data values. To solve this, current methods calculate numerical

[1]https://vega.github.io/vega-lite



```
{
"mark":{"type": "bar"},
"encoding": {
"x": {
"scale": {"domain": ["All adults", "Age 18-29",
"Age 30-49", "Age 50+"] , "type": "ordinal" }},
"y":{
"type": "quantitative",
.......
}
```

	Nonbinary	Trans man/woman
All adults	0.75	0.5
Age 18-29	2.5	2.25
Age 30-49	1.5	0.75
Age 50+	0.3	0.5

```
{
"mark":{"type": "bar"},
"encoding": {
"x": { .......
}
```

	Nonbinary	Trans man/woman
All adults	1.0	0.6
Age 18-29	3.0	2.0
Age 30-49	1.3	0.3
Age 50+	0.1	0.2

Some 3.0% of adults younger than 30 are nonbinary.

Age	Nonbinary	Trans man/woman
All adults	1.0	0.6
Age 18-29	3.0	2.0
Age 30-49	1.3	0.3
Age 50+	0.1	0.2

eq{hop{filter_eq{Age; Age 18-29}; Nonbinary}

datum.Age == "Age 18-29" && datum.group == "Nonbinary"

```
{"mark":{"type": "bar"},
"encoding": {
.......
"transform":[{
    "Filter":"datum.Age == 'Age 18-29'
    && datum.group == 'Nonbinary'"},
    .......
```

Figure 2: Given a chart image, HiChart follows three steps: (1) reverse-engineering the visualization process to generate a visualization specification S, (2) calibrating the data in S such that the generated visualization is closer to the original chart image, and (3) highlighting the parts in the chart that are relevant to a text span T provided by a user.

values in a chart by detecting the key points [4] or bounding boxes of graphic components [3]. However, in this case, unless bounding boxes perfectly match the shapes of graphic components, which is hard to achieve, the calculated value will be inaccurate.

To address this problem, we propose a novel method to tune the extracted data values for visualization calibration. Specifically, suppose that we have already recovered the specification S and dataset D' from a chart image V, while D' is different from D. For calibration, we tune the initial extracted data via *value perturbation* based on a similarity measurement between the generated and the original images, as shown in *Step 2* of Figure 2, which is similar to forward-propagation in machine learning. The extracted data and similarity between charts can be regarded as the initial state and the objective function respectively. To achieve this goal, we need to address two main challenges, namely measuring similarity between charts and tuning data based on the similarity changes.

In general, measuring the similarity between two similar chart images is challenging because it should only measure the difference between *data* contained in the charts and be very sensitive to the pixel-level changes of graphic components which reflect data values. Currently, HiChart considers the case that the chart image V is generated by the same language of S, which makes it easier to compare the original chart with the generated one. Vega-Lite grammar is used to generate chart images and parameters like font size and padding keep fixed in the specifications. Then, we use a typical hashing algorithm to encode the chart images and compute similarities based on the encoded results. We find that such similarity measurement can measure the pixel-level change in a chart, especially being very effective for bar charts.

For data tuning, we need to update the extracted data values based on the change of similarity between the original and the re-generated charts. As it is hard to know the exact gap between the original and extracted data directly, we perform *value perturbation* on the initial value, *e.g.,* increasing or decreasing it by a step value, and then re-generate a chart. For instance, if the similarity between them becomes smaller regardless of the value plus or minus the step value, the step value will be reduced. If the similarity between the re-generated chart and the original one becomes smaller, it means that the updated value is closer to the ground truth. For example, as shown in *Step 2* of Figure 2, the cell value can be changed from *0.75* to *1.0.* The above process will be repeated until a converging condition is met, *e.g.,* similarity is smaller than a threshold. In particular, inspired by the gradient descent training process, we dynamically update the step value for fast convergence.

To verify this method's effectiveness, we have conducted a preliminary experiment. We select 1111 bar charts from the dataset in [5] and use the same evaluation metric as [5]. After adopting the data tuning method, when the acceptable error is within 1%, the accuracy increases from 0.089 to 0.463; when the acceptable error is within 5%, the accuracy increases from 0.674 to 0.884.

2.3 Chart Highlighting for Text Span

Given a span of text T provided by a user, chart highlighting identifies the parts of a chart corresponding to T, and then augments specification S with highlighting scripts. The limitation of most related works is that they cannot support natural language processing (NLP) based chart highlighting. While some recent approaches support highlighting charts given text input [6], the approaches typically require that text input follows pre-defined templates, and thus they are inadequate for flexible chart highlighting.

To address the problem, HiChart utilizes the Latent Program Algorithm (LPA) [1] to perform semantic parsing for the text span in chart highlighting. The framework is shown in *Step 3* of Figure 2, where the input includes the chart and its specification S, dataset D of the chart and a text span T.

Given text span T and dataset D, LPA identifies linked entities and keywords in the text span, and uses the keywords to find the operations that can be used. Then, it generates potential operations on the entities, and these operations compose a tree-like program, which is a semantic representation of the text span, as shown in Figure 2. Then, we use the margin loss as a semantic similarity score to select the most relevant program from candidate programs.

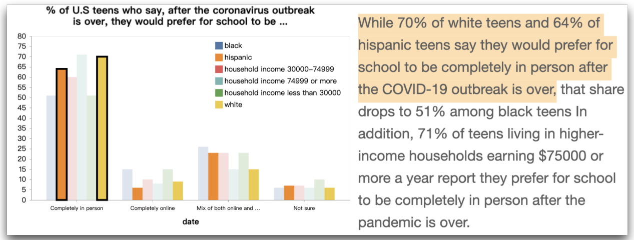
Next, we translate the selected program into Vega-Lite code by using some pre-defined templates, which is then augmented to the original specification S. Take the text in Figure 2 as example: the generated program is converted to a part of Vega-Lite code for highlighting effect. Finally, the generated code is added into specification S, as illustrated in *Step 3* of Figure 2. For the cases that program generation is not successful, we augment specification S based on the matched entities. After we obtain the specification related to highlighting, we will make changes to the induced specification S in the previous step, including adding key information related to the chart, adding highlighting code and so on. In particular, HiChart supports chart highlighting for three types of charts, namely pie chart, line chart and bar chart.

3 DEMONSTRATION SCENARIOS

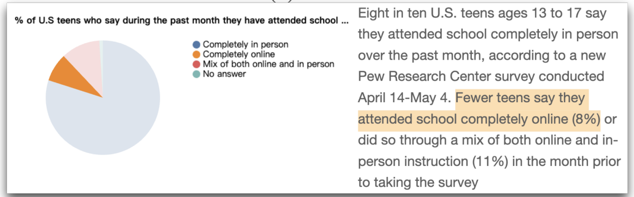
We present a web application to all participants and illustrate HiChart in the following chart highlighting scenarios (see https://youtu.be/3I5FEND3brE for a video). We use a dataset that contains raw chart data and text from news published by the PEW research center. The news we use covers a variety of topics. Figure 3 shows three examples supported by HiChart. In particular, we re-generate these chart images in Vega-Lite grammar based on raw chart data, as most charts from the web are not standardized and vary a lot. For example, the chart in Figure 1 would be re-generated as a standard grouped bar chart, which would facilitate easier processing.

Extracting web charts with intuitive results. In the demo, a user can enter a news URL link containing text (*e.g.,* news paragraphs) as well as visual charts, and then click "GO" to go to the corresponding page. Then, the user can select the paragraph she wants to analyze, along with the chart that corresponds to the text. After selecting the chart, HiChart extracts information to induce the specification of this chart and obtain chart-related data. Thus, the user can select the new displayed chart and download the chart data if he wants.

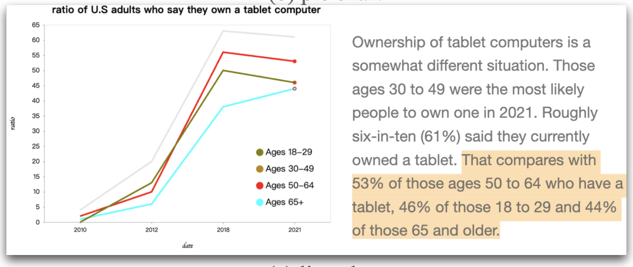
Sentence based chart highlighting. When the user chooses a paragraph, HiChart analyzes each sentence from the paragraph and generates highlighting result if the sentence is related to the chart. For bar and pie charts, HiChart highlights the relevant part and shadow the other irrelevant ones, which can be seen in Figures 3 (a) and (b). For the bar chart (a), we highlight the groups of White and Hispanic involved in the sentences, and emphasize the columns of data of those preferring to attend school completely in person in these two groups. In this way, users can quickly compare data mentioned in sentences and get information from charts without spending time estimating the correspondence of values in charts or looking for labels for data in sentences. In Figure 3 (c), it can be seen that when highlighting line charts, HiChart highlights not only the lines in which the relevant data is grouped but also the data points involved in the text span.


% of U.S teens who say, after the coronavirus outbreak is over, they would prefer for school to be ...
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
black
hispanic
household income 30000–74999
household income 74999 or more
household income less than 30000
white
Completely in person
Completely online
Mix of both online and ...
Not sure
date
While 70% of white teens and 64% of hispanic teens say they would prefer for school to be completely in person after the COVID-19 outbreak is over, that share drops to 51% among black teens In addition, 71% of teens living in higher-income households earning $75000 or more a year report they prefer for school to be completely in person after the pandemic is over.

(a) bar chart


% of U.S teens who say during the past month they have attended school ...
Completely in person
Completely online
Mix of both online and in person
No answer
Eight in ten U.S. teens ages 13 to 17 say they attended school completely in person over the past month, according to a new Pew Research Center survey conducted April 14-May 4. Fewer teens say they attended school completely online (8%) or did so through a mix of both online and in-person instruction (11%) in the month prior to taking the survey

(b) pie chart



Ownership of tablet computers is a somewhat different situation. Those ages 30 to 49 were the most likely people to own one in 2021. Roughly six-in-ten (61%) said they currently owned a tablet. That compares with 53% of those ages 50 to 64 who have a tablet, 46% of those 18 to 29 and 44% of those 65 and older.

(c) line chart

Figure 3: Highlight results of three chart types.

ACKNOWLEDGMENTS

This work was partly supported by the NSF of China (62122090 and 62072461), the Special Funds for Central Universities to Build World-class Universities (Disciplines) and Guide the Development of Distinctive Features - Basic Research Funds for Central Universities of Renmin University of China, 2022.

REFERENCES

[1] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *ICLR*.
[2] Chunxiao Liu, Zhendong Mao, Tianzhu Zhang, Hongtao Xie, Bin Wang, and Yongdong Zhang. 2020. Graph Structured Network for Image-Text Matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[3] Xiaoyi Liu, Diego Klabjan, and Patrick NBless. 2019. Data extraction from charts via single deep neural network. *arXiv preprint arXiv:1906.11906* (2019).
[4] Junyu Luo, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. 2021. ChartOCR: data extraction from charts images via a deep hybrid framework. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 1917–1925.
[5] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244* (2022).
[6] Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A Natural Language Interface for Visual Analysis. In *UIST*. ACM, 365–377.
[7] Fangfang Zhou, Yong Zhao, Wenjiang Chen, Yijing Tan, Yaqi Xu, Yi Chen, Chao Liu, and Ying Zhao. 2021. Reverse-engineering bar charts using neural networks. *Journal of Visualization* 24, 2 (2021), 419–435.